Universidad Zaragoza

Trabajo Fin de Grado

Gestión de la distribución del tráfico en un entorno WLAN

Management of traffic distribution in a WLAN environment

Autor

Pedro Antonio Fortón Rubio

Director

José Mª Saldaña Medina

Ponente

Julián Fernández Navajas

ESCUELA DE INGENIERIA Y ARQUITECTURA

2019

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. Pedro Antonio Fortón Rubio,

con nº de DNI 73027297C en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Grado, (Título del Trabajo) Gestión de la distribución del tráfico en un entorno WLAN,

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 30/01/2019

Fdo: Pedro Antonio Fortón Rubio

Agradecimientos

Mi más sincero agradecimiento a mi familia y mis amigos por su apoyo todos estos años de formación. En especial a mis padres por estar conmigo en los momentos complicados y por hacer posible que haya podido llegar hasta aquí.

También quiero agradecer a todos mis compañeros de viaje durante esta etapa de mi vida, a todas aquellas personas que me han ayudado incondicionalmente cuando ha sido necesario, profesores, compañeros y amigos.

Y no quiero olvidarme de la labor que han llevado a cabo José Mª y Julián, orientándome y ayudándome en este proyecto, mi más sincero agradecimiento.

Resumen

Actualmente el mundo es un lugar conectado, en el que las redes de la información tienen una presencia continua en nuestra vida cotidiana. Una forma de acceder a estas redes es mediante el uso de redes Wi-Fi. Están presentes en muchos hogares, pero también existen soluciones que integran varios puntos de acceso coordinados, para dar servicio a una zona (aeropuertos, universidades, centros de negocios, etc.).

El proyecto H2020 Wi-5, haciendo uso de tecnologías SDN (Software Defined Networking, Redes Definidas por Software), propuso múltiples soluciones a esta problemática. El proyecto realizó y probó diferentes propuestas para mejorar el servicio que ofrecen las redes Wi-Fi a través de puntos de acceso coordinados.

Una vez finalizado el proyecto, se desea seguir añadiendo nuevas funcionalidades no consideradas al inicio del proyecto. Se presenta la problemática de que algunas de las soluciones generadas resultan poco amigables para el usuario, y poco mantenibles a largo plazo. Por ejemplo, algunas soluciones presentan los datos de una manera que puede resultar confusa y con carácter únicamente informativo.

El presente proyecto afronta el problema proponiendo una interfaz gráfica usable y amigable, que presente los datos en tiempo real y permita interacción con el sistema. Entre las múltiples aplicaciones desarrolladas por el proyecto H2020, se elige desarrollar dicha interfaz gráfica para la aplicación más completa de todas, *Smart AP Selection.* Esta aplicación combina los resultados de varias aplicaciones desarrolladas durante el proceso del proyecto H2020, como es la gestión de la movilidad, o el balanceo de carga.

Se plantea la aplicación como un componente más del proyecto Wi-5, complementando los desarrollos anteriores, añadiendo nuevas funcionalidades, sin realizar cambios esenciales. Durante el desarrollo de este Trabajo Fin de Grado se ha analizado, diseñado y implementado un cliente ligero, denominado Odin-Wi5Gui, que muestra de una manera accesible y usable la compleja información que genera un sistema como es Wi-5.

De manera complementaria se han realizado tareas de mantenimiento sobre los proyectos anteriores, aprovechando esta oportunidad para mejorar el proyecto, dotándolo de una mayor escalabilidad y facilidad de mantenimiento. El proyecto se centra en el desarrollo de la aplicación de escritorio Odin-Wi5Gui, condicionando las decisiones de diseño derivadas de las expectativas de posibles futuras mejoras. De esta forma, el sistema actualizado puede usarse como un escalón a una aplicación mucho más compleja, pero a su vez mantenible y usable.

Índice

Glosario

- Estación (STA): Dispositivo capaz de conectarse a una red mediante el protocolo 802.11 (conocido como Wi-Fi), por ejemplo, un móvil, un portátil o una tablet. Se denominara también “cliente”.
- Punto de acceso (Access Point, abreviado AP): dispositivo de red físico capaz de interconectar estaciones, formando redes inalámbricas, como por ejemplo una red WLAN.
- MoSCoW. Técnica de priorización utilizada en la gestión, análisis de negocios, gestión de proyectos y desarrollo de software para llegar a un entendimiento común con las partes interesadas sobre la importancia que otorgan a la entrega de cada requisito. El término MoSCoW es un acrónimo derivado de la primera letra de cada una de las cuatro categorías de priorización (*Must have, Should have,Could have*, y *Won't have*).
- DHCP (*Dynamic Host Configuration Protocol*, Protocolo de configuración dinámica de host): es un protocolo de red de tipo cliente/servidor que asigna dinámicamente una dirección IP a los clientes conectados a la red.
- RSSI (*Received Signal Strength Indicator)*: El indicador de fuerza de la señal recibida por un dispositivo en una red inalámbrica. Este parámetro indica la potencia de la señal, no su calidad.
- HTTP (*Hypertext Transfer Protocol,* Protocolo de transferencia de hipertexto): Protocolo a nivel de aplicación para sistemas de la información ya sean distribuidos, colaborativos o sistema de hypermedia.
- Base de datos NoSql. (*Non Relational):* Este tipo de bases de datos se caracteriza por proporcionar un mecanismo de almacenamiento y recuperación de los datos almacenados de manera no relacional.
- Dirección IP (*Internet Protocol Address):* Número identificativo que se asigna a cada dispositivo que usa el *Internet Protocol*. Las direcciones IPv4 se componen de número de 32 bits y las direcciones Ipv6, fueron desarrolladas posteriormente y se componen de números de 128 bits.
- JSON (*JavaScript Object Notation):* Formato de ficheros que transmite objetos consistentes en pares de datos clave-valor y *array*s de datos, mediante el uso de lenguaje “humano”.
- MVC (*Models - View - Controller):* Patrón de arquitectura de software compuesto por tres componentes, modelos, vistas y controladores. Este patrón permite separar la lógica de una aplicación de la lógica de la vista.
- CRUD (*Create, read, update and delete):* Acrónimo de las funciones básicas, (crear, leer, actualizar y eliminar) de la información almacenada en un sistema informático.

Capítulo 1 - Odin Wi-5Gui

1.1 Introducción

El desarrollo del presente Trabajo Fin de Grado (TFG) se sitúa dentro de la línea de investigación iniciada con el proyecto europeo H2020 “What to do With the Wi-Fi Wild West” (Wi-5) [1], que terminó en abril de 2018. En dicho proyecto se estudiaron e implementaron soluciones SDN (Software Defined Networks, Redes Definidas por Software) para gestionar y coordinar los distintos puntos de acceso (Access Points, AP) de una red inalámbrica mediante software libre.

El proyecto Odin Wi-5Gui busca mejorar el software desarrollado por el equipo de la Universidad de Zaragoza que participó en el proyecto Wi-5. Se desarrollaron varias aplicaciones para coordinar un número de puntos de acceso Wi-Fi desde un controlador central, y que son capaces de realizar funciones como la gestión de la movilidad de los usuarios (cambiarlos dinámicamente de punto de acceso), el balanceo de carga (buscar un equilibrio entre el número de usuarios que están conectados a cada punto de acceso) o la asignación óptima de canales a los AP.

Los objetivos de Wi-5Gui nacen de la necesidad de una mejor explotación de los datos generados por el conjunto de aplicaciones desarrolladas por el equipo. Durante el proyecto Wi-5 no se planteó como requisito el desarrollo de una interfaz amigable que permitiera la monitorización del estado de la red, o el análisis de la información generada por el sistema. La información se mostraba mediante trazas en la consola del equipo controlador de la aplicación, lo que impedía una correcta explotación de los datos generados por el sistema.

El presente Trabajo Fin de Grado tiene por tanto el principal objetivo de mejorar la presentación de la información, para permitir un mejor control y por tanto, facilitar la explotación de la información disponible.

El trabajo se centra en el desarrollo de dos mejoras para el software del proyecto Wi-5:

- El diseño y desarrollo de una interfaz gráfica amigable que permita al administrador de la red la monitorización y análisis de datos en tiempo real.
- La optimización y depuración de código actual, con el objetivo de aumentar la escalabilidad del proyecto a futuro.

Se han estudiado diferentes alternativas para alcanzar los objetivos deseados, cubriendo todas las necesidades. Como conclusión de un análisis preliminar, se ha optado por desarrollar estas mejoras siguiendo los principios de las metodologías ágiles, que permitirá al desarrollador afrontar los problemas planteados y llevar a cabo las soluciones acordadas con el grupo de investigación que trabajó anteriormente en el proyecto.

El resultado del Trabajo Fin de Grado es elaborar mejoras para el software Wi-5, con el objetivo de aumentar la escalabilidad, robustez y mantenimiento a largo plazo del proyecto al completo, eligiendo tecnologías y arquitecturas pensando en las exigencias futuras. Y dando por supuesto que este software podrá ser continuado y mejorado en futuros proyectos con el grupo de investigación que participó en Wi-5, o en la realización de nuevos TFGs.

1.2 Objetivo del trabajo fin de grado

Los objetivos de este proyecto se pueden ver como una continuación del trabajo desarrollado en el Work Package 3 del proyecto Wi-5, titulado *Smart AP Solutions*, liderado por el equipo de la Universidad de Zaragoza. Como parte de este trabajo se desarrolló una aplicación *(Smart AP Selection [2])* que realiza un escaneo constante del entorno inalámbrico, para así realizar traspasos entre puntos de acceso cuando es necesario. De esta manera, se puede buscar el mejor AP para cada dispositivo, y distribuir óptimamente el tráfico entre los AP disponibles. Para realizar correctamente estas funciones, es necesario disponer de la información en tiempo real sobre el estado de la red. La forma que tiene de mostrar la información es a partir de una aplicación de consola sin persistencia de ninguna clase, lo que produce un alto grado de dificultad a la hora de obtenerla y realizar análisis con los datos obtenidos.

A partir de estas necesidades, el presente Trabajo Fin de Grado nace para actualizar la aplicación de consola, convirtiéndola en una aplicación amigable para el usuario, capaz de extraer, tratar y documentar los datos obtenidos a lo largo de la ejecución del programa principal, facilitando así las tareas de análisis y monitorización de la red donde se aloje la aplicación *Smart AP Selection*.

Como punto de partida para el proyecto se han marcado los siguientes objetivos principales:

- Creación de una interfaz gráfica amigable en tiempo real, que permita la monitorización de la aplicación *Smart AP Selection*.
- Análisis y tratamiento de toda la información generada por la aplicación *Smart AP Selection,* mostrada de manera gráfica y sencilla.
- Disponer de una API donde obtener toda la información del sistema en tiempo real.
- Permitir cambios en tiempo real en la red física.
- Refactorización del proyecto permitiendo una mayor escalabilidad y robustez.
- Dar persistencia de alguna clase a los datos obtenidos durante la ejecución de la aplicación.
- Permitir que el usuario pueda elegir los parámetros de la aplicación que se está ejecutando y modificarlos en tiempo real.
- Optimizar el código existente con el objetivo de reducir el tiempo de ejecución de la aplicación *Smart AP Selection*.
- Introducir un sistema de log exclusivo de la aplicación *Smart AP Selection*.

1.3 Alcance

Antes de definir el alcance que debe tener el proyecto actual, se ha realizado un análisis preliminar sobre las necesidades (Capítulo 2) presentadas por el grupo que trabajó en el proyecto Wi-5. Las conclusiones obtenidas a partir del análisis dictan que, si se implementaran y desarrollaran todas las posibles mejoras planteadas, se superaría el trabajo estimado para un TFG. Partiendo de este punto, este proyecto se realiza a partir de la idea de que algunas de las peticiones no puedan cumplirse. Este proyecto se ve como un punto de partida para el desarrollo de futuras mejoras que puedan complementarlo.

Gracias a un estudio previo de las necesidades presentadas, el análisis de los datos que genera la aplicación *Smart AP Selection* (Sección 2.1), un primer acuerdo de requisitos (Sección 2.3) y un análisis de riesgos y viabilidad (Sección 2.4), se ha decidido que el alcance de este trabajo se centrará en la creación de una interfaz gráfica en tiempo real, y en la depuración y optimización de código presente en *Smart AP Selection*, permitiendo así una mayor escalabilidad y mantenimiento a largo plazo.

Se debe considerar que actualmente el software de Wi-5 se sigue mejorando, aumentando el volumen de código y tiempo invertido en él. Uno de los objetivos secundarios de proyecto es centrar esfuerzos en la generación de código limpio y robusto que permita en un futuro insertar nuevas funcionalidades sin un gran esfuerzo por parte del desarrollador.

1.4 Estructura del documento

El documento actual se ha organizado con la siguiente disposición:

- Capítulo 1. Descripción del proyecto, objetivos y alcance de Trabajo Fin de Grado.
- Capítulo 2. Análisis del proyecto, obtención de requisitos finales y análisis de las posibles soluciones y sus riesgos.
- Capítulo 3. Diseño de los requisitos funcionales presentados con anterioridad, presentando los problemas que afrontan y sus soluciones.
- Capítulo 4. Se detallan los pasos para el desarrollo e implementación de las soluciones presentadas en el capítulo de diseño.
- Capítulo 5. Se detalla la organización, gestión y planteamiento del proyecto.
- Capítulo 6: Se exponen las conclusiones del trabajo.
- Anexos.

Capítulo 2- Análisis y requisitos del sistema

En este apartado se resume el análisis realizado inicialmente para definir los requisitos del sistema, a partir del software disponible y de las mejoras requeridas. Posteriormente a la toma de decisiones en el diseño del proyecto, se ha estudiado la situación actual del software (Sección 2.1), realizando un análisis global de todo el código desarrollado durante el proyecto Wi-5, y especialmente de la aplicación *Smart AP Selection*, con el objetivo de identificar potenciales problemas que puedan surgir a lo largo del desarrollo del proyecto.

Se ha elaborado un primer documento funcional (Sección 2.2), siguiendo la metodología MoSCoW [3], permitiendo obtener las necesidades del grupo de investigación y considerar si son asumibles por este Trabajo Fin de Grado. Complementando este documento, se ha realizado un análisis de viabilidad (Sección 2.3) de las diferentes alternativas presentadas para llevar a cabo el Trabajo Fin de Grado, realizando una prueba de concepto que cumpla los requisitos funcionales. Tras estos dos análisis se han estudiado los posibles riesgos que podrían aparecer a lo largo del proyecto (Sección 2.4). Y, por último, se han definido los requisitos finales (Sección 2.5) que deben estar presentes en la solución final.

2.1 Descripción de la situación inicial

2.1.1 Elementos del sistema

Se parte de la solución Odin-Wi-5 [4], que se desarrolló como una extensión de la plataforma Odin [5], presentada en [6] que se compone de varios elementos (ver Fig. 2.1):

- Un controlador *(Controller VM* en la figura), implementado como un módulo para el *Floodlight OpenFlow Controller [7]*. El grupo de investigación de la universidad de Zaragoza desarrolló un conjunto de aplicaciones *(Smart Functionalities)* que permitan balancear la carga, gestión de la movilidad, obtención de estadísticas de la red en tiempo real, etc.
- Un conjunto de AP, que son gestionados por el controlador, y que incluyen funciones de monitorización, traspasos suaves entre APs (*seamless handoff),* etc. Los parámetros monitorizados son enviados al controlador, que es quien toma las decisiones en función del estado de la red.
- La red contiene un router que actúa de servidor DHCP (*server VM* en la figura).

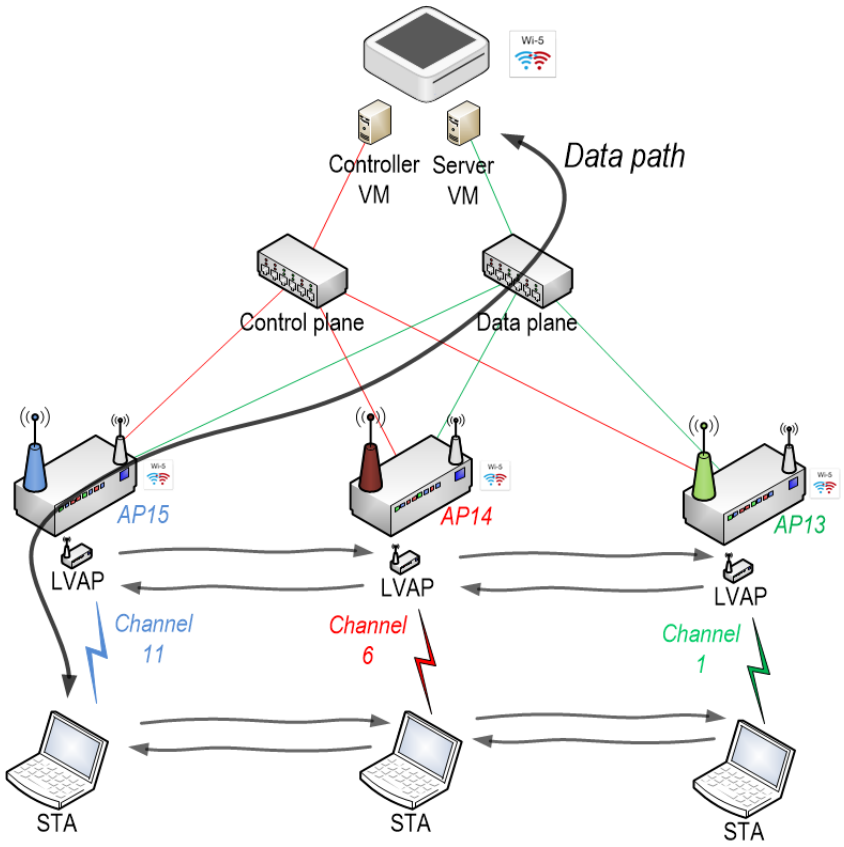

Fig. 2.1 Elementos del sistema Odin-Wi-5

Aunque el equipo desarrolló múltiples aplicaciones que pueden correr sobre *Odin*, para este Trabajo Fin de Grado se ha solicitado añadir una interfaz gráfica a la aplicación más completa de todas, *Smart AP Selection* [8]. Esta aplicación, utiliza un método proactivo para realizar una gestión de la movilidad de los usuarios (asignar cada dispositivo al AP que resulte óptimo), y simultáneamente un balanceo de carga (conseguir una distribución equilibrada entre los AP) [9]. Mediante el protocolo *Odin*, solicita de manera periódica a los diferentes puntos de acceso de la red un escaneo en sus respectivos canales. (en el Anexo 1 puede encontrarse la información que generan dichos escaneos). La información obtenida en tiempo real permite al controlador, mediante diferentes algoritmos, la asignación de las estaciones conectadas a la red, según la posición en la que se encuentren *(mobility management)*, así como distribuir óptimamente el tráfico entre los AP disponibles *(load balancing)*. El tiempo empleado en la iteración principal debe ser lo suficientemente breve como para permitir al controlador tomar decisiones de traspaso entre puntos de acceso para usuarios que caminan. En las pruebas realizadas durante el proyecto Wi-5, se mostró que estas decisiones se deben tomar cada 2 segundos como máximo [10].

2.1.2 Aplicación *Smart AP Selection*

Tras la correcta ejecución, configuración y lanzamiento del sistema, se puede lanzar la aplicación *Smart AP Selection,* que muestra por pantalla información en tiempo real de los clientes conectados y los APs respectivos (Fig 2.2), informando de la aplicación que se está ejecutando en el momento, así como la información de los posibles clientes conectados.

Como se puede apreciar en la Fig. 2.2, esta pantalla es solamente informativa, no permite ninguna interacción con el usuario y carece de información histórica. Otra limitación que se encuentra es que esta pantalla está siendo compartida por todos los servicios y módulos del controlador, pudiendo escribir sobre ella las múltiples trazas informativas que puede generar un controlador tan complejo como *Floodlight*.

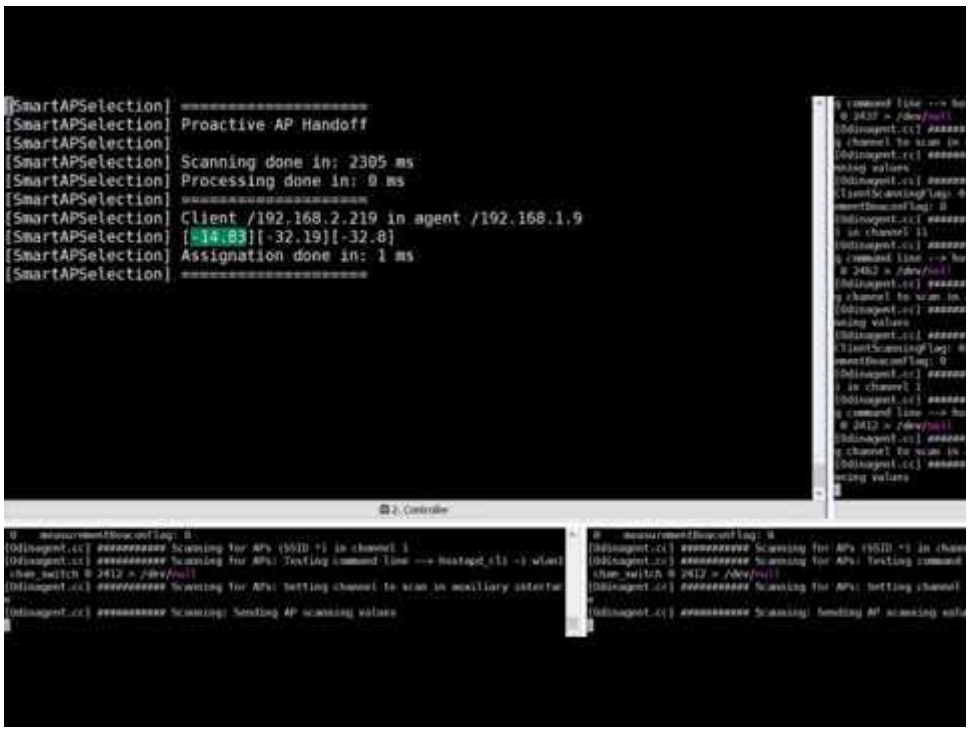

Fig. 2.2 Salida de información de la aplicación inicial *Smart AP Selection*

Por tanto, gran parte de la información que genera el sistema no se puede mostrar actualmente de una manera eficaz y amigable para el usuario. Sin embargo, otra de las aplicaciones desarrolladas, *DemoStatistics* [11], sí permite al usuario una mayor comprensión de la red actual (ver Anexo 1 - información de *DemoStatistics)*.

Una de las estadísticas que se pueden obtener es la *matriz de atenuación,* es decir, una matriz que muestra la atenuación de la señal entre cada punto de acceso y las estaciones. Cuanto más lejos estén la estación de su punto de acceso, mayor será la atenuación, y habrá más interferencia entre ellos.

2.1.3 Limitaciones detectadas

Algunas de las limitaciones encontradas en la aplicación son las siguientes:

- La interfaz gráfica sólo es accesible desde la terminal de administrador donde se lanza el programa *principal*, impidiendo con ello la posibilidad de analizar la red de manera remota e independientemente al programa principal.
- Las trazas realizadas por Odin y los diferentes subsistemas que lo componen también escriben sobre la misma pantalla, “ensuciando” con ello las trazas del programa principal desarrollado por Wi5, el cual muestra los datos que se consideran esenciales.

- Otro problema detectado es la nula separación de la lógica de *Smart AP Selection* de la actual interfaz gráfica: la capa de lógica imprime por pantalla según se ejecuta, "ensuciando" el código y añadiendo posibles riesgos de fallo en una lógica que debería funcionar independientemente de la interfaz gráfica.
- El código existente se planteó como una aplicación monolítica, produciendo con ello las limitaciones que poseen dicho tipo de arquitecturas: código difícil de mantener, nula reutilización de código, etc. El conjunto de aplicaciones disponibles poseía numerosas líneas de código repetido y se ha necesitado tiempo para comprender la funcionalidad principal de la aplicación.

2.2 Obtención de posibles requisitos

2.2.1 Metodología

Como primera aproximación para la captura de requisitos del sistema se ha utilizado como referencia la metodología MoSCoW. El método MoSCoW es una técnica de priorización de requisitos basada en el hecho de que, aunque todos los requisitos se consideren importantes, es fundamental destacar aquellos que permiten proporcionar un mayor valor al sistema, lo que permite enfocar los trabajos de manera más eficiente. Lo que la diferencia de otras técnicas tradicionales, como por ejemplo calificar los requisitos como de prioridad alta, media o baja, es que en este caso la escala utilizada tiene un significado intrínseco, de manera que el usuario responsable de asignar la prioridad conoce el efecto real que produciría su elección. La escala se compone de los siguientes niveles:

- ***Debe***: Requisito que tiene que estar implementado en la versión final del producto para que la misma pueda ser considerada un éxito.
- ***Debería***: Requisito de alta prioridad que, en la medida de lo posible, debería ser incluido en la solución final, pero que, llegado el momento y si fuera necesario, podría ser prescindible si hubiera alguna causa que lo justificara.
- ***Podría***: Requisito deseable pero no necesario, que se implementaría si hubiera posibilidades presupuestarias y temporales.
- ***No hará***: Hace referencia a requisitos que están descartados de momento, pero que en un futuro podrían ser tenidos de nuevo en cuenta y ser reclasificados en una de las categorías anteriores. Estos requisitos podrían ser abordados en futuros desarrollos del proyecto.

Esta clasificación puede ser modificada durante el proceso de desarrollo y permite definir, en el caso de desarrollos iterativos incrementales, prioridades a nivel de iteración.

2.2.2 Entrevista con el encargado del desarrollo - Captura de primeros requisitos

Usando el método MoSCoW como punto de partida se realizó una entrevista al encargado del proyecto donde, de manera casual se definieron algunos de los requisitos funcionales presentes. La prioridad asignada a cada requisito se realizó asumiendo que no todas las necesidades actuales del proyecto son asequibles para un Trabajo Fin de Grado. Esta es la primera clasificación de requisitos que se realizó:

Al final de este TFG, el proyecto ***debe*** ser capaz, de:

1. Recolectar y ofrecer en tiempo real información sobre el estado del sistema y los diferentes dispositivos conectados al mismo.
2. Ofrecer la matriz de atenuación de la red.
3. Ofrecer la posibilidad de solicitar el cambio de canal de un punto de acceso.
4. Ofrecer la posibilidad de solicitar el "handoff" (traspaso) de un dispositivo de su actual punto de acceso a otro presente en el sistema.
5. Ofrecer la información de todos los dispositivos conectados actualmente.
6. Ofrecer información obtenida del escaneo de un canal por uno o varios puntos de acceso.
7. El proyecto será analizado y refactorizado respetando la arquitectura en tres capas, capa de presentación, capa negocio y capa de datos, permitiendo una mejora de escalabilidad y mantenimiento.
8. El proyecto será desarrollado en Java y poseerá una arquitectura monolítica.

El resultado del proyecto ***debería*** ser capaz, al final de este TFG, de:

1. Ofrecer gráficas representativas del estado del sistema.
2. Realizar la iteración principal de la aplicación *Smart AP Selection* en menor o igual tiempo que el requerido por la versión inicial, en este caso dos segundos.

El resultado del proyecto ***podría*** ser capaz al final de este TFG:

1. Asignar puntos de acceso como "VIP" (es decir, reservados para algunos usuarios privilegiados).
2. Obtener información de los flujos de *Open vSwitch* (software que se utiliza en cada punto de acceso para controlar qué estaciones están conectadas).
3. Iniciar o modificar la aplicación que actualmente está ejecutándose en *Odin-Wi-5*.

El resultado del proyecto ***no hará*** al final de este TFG:

1. Se podrá modificar el sistema para asignar nuevas reglas para los puntos de acceso "VIP".
2. Se podrá añadir/eliminar puntos de acceso al sistema durante la ejecución del mismo.

2.3 Estudio de requisitos y su viabilidad

Como punto de partida en el estudio de requisitos, se han ponderado diferentes tecnologías y arquitecturas que posibiliten la solución. Durante el transcurso de éste, se ha llevado a cabo un estudio exhaustivo del sistema sobre el que trabaja (Sección 2.3.4), permitiéndole tomar mejores decisiones en función del trabajo ya existente. Este estudio finaliza con una prueba de concepto (Sección 2.3.3) que permite cumplir los requisitos considerados principales, y la definición de requisitos finales en función de los resultados.

2.3.1 Análisis y refactorización del proyecto

En este punto se ha llevado a cabo el análisis de la aplicación *Smart AP Selection*, con el fin de comprender el código, para poder refactorizarla sin afectar a su rendimiento, manteniendo así el requisito funcional número 2 de los requisitos catalogados como ***debería***.

La interacción principal de la aplicación *Smart AP Selection* se compone de una única clase de 1179 líneas de código, donde la función principal ocupa unas 500 líneas de código [12]. La aplicación posee diferentes modos de ejecución, todos ellos bajo la misma función.

La función principal realiza, en el siguiente orden, las siguientes acciones:

- Inicializa las variables que se utilizarán a lo largo de toda la ejecución.
- Solicita un escaneo activo a todos los AP conectados al controlador, solicitando el nivel de RSSI (*Received Signal Strength Indicator,* el indicador de fuerza de la señal recibida) recibido de cada una de las estaciones conectadas.
- Con la información obtenida a través de cada una de las estaciones, la función crea y almacena una matriz de atenuación. Haciendo uso de esta matriz, se organizará la red en busca de una configuración óptima.
- Una vez obtenida la matriz, la función actualiza en pantalla los datos presentados al usuario, mostrando en tiempo real el estado de la red.
- Por último, en la parte final de la aplicación se encuentran los diferentes modos de que dispone *Smart AP Selection.* Los parámetros de configuración permiten al usuario indicar cuáles de los modos se ejecutarán.

Aunque actualmente la función principal posee pocas líneas de código, si se sigue desarrollando sobre esta misma aplicación es posible que los desarrolladores se encuentren con múltiples problemas de mantenibilidad. El objetivo de este análisis preliminar es reducir ese número de líneas a un número aceptable, mejorando la mantenibilidad y lectura del código, aplicando a ser posible los principios SOLID [13].

El análisis preliminar también tiene por objetivo encontrar una manera de obtener los datos generados por la aplicación sin detener o demorar su ejecución. Los datos deben encontrarse disponibles de manera asíncrona, de forma que la interfaz gráfica consuma dichos datos sin detener la ejecución de la función principal. Para la resolución de este problema se desarrolla una pequeña clase que actúa de almacén de datos asíncrono, permitiendo al proceso de la interfaz gráfica acceder a los datos producidos por la función principal. Esta clase va a ser atacada por diferentes procesos, por tanto, debe tenerse en cuenta los posibles problemas de concurrencia que podrían producirse.

2.3.2 Interfaz Java

El requisito técnico número 8 plantea la siguiente limitación a la hora de elegir una tecnología adecuada para este proyecto: debe realizarse en Java. Este requerimiento viene impuesto por el grupo de investigación, pues el software del controlador Wi-5 está prácticamente desarrollado sobre Java y el grupo de investigación prefiere que todos los futuros desarrollos se realicen en este lenguaje.

Se presentan múltiples herramientas con las que realizar esta interfaz, AWT, Swing, SWT, JavaFx, etc. Para la prueba de concepto se utilizará un framework Java nativo como es Swing,

respetando así el requisito número 8. Esta tecnología, aunque posee ciertas limitaciones, cuenta con mucha documentación y el estudiante posee experiencia con ella.

La forma de mostrar el estado de la red mediante una matriz de atenuación entre estaciones y puntos de acceso puede resultar poco amigable para un usuario no familiarizado con el sistema. Por tanto, se opta por mostrar dicho estado de una forma visual y gráfica. Con esos requerimientos se realiza un estudio de las posibles tecnologías compatibles con Java que permitan cumplir estas necesidades. Se encuentra la librería GraphStream [14], desarrollada en Java, que permite mostrar grandes cantidades de datos mediante grafos en tiempo real (Fig. 2.3).

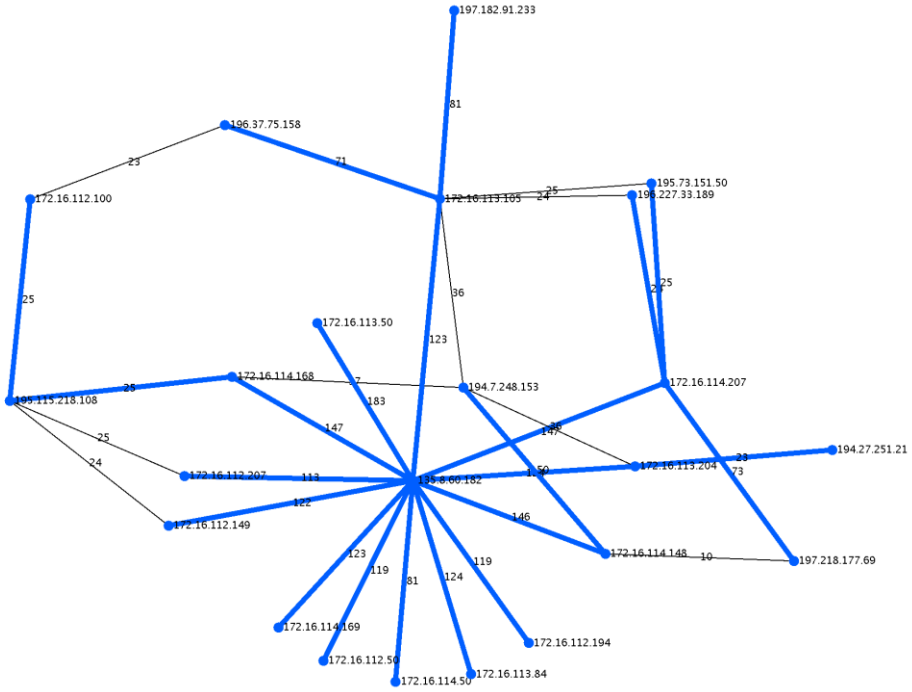

Figura 2.3. Ejemplo de una red dibujada mediante la librería *GraphStream*

2.3.3 Prueba de concepto

Para el estudio de los requisitos planteados anteriormente, se ha llevado a cabo una prueba de concepto con el objetivo de comprobar si los requisitos funcionales de mayor peso podrían cumplirse con las tecnologías elegidas (2.3.1 y 2.3.2) y la arquitectura planteada (requisito número 9).

Los requisitos por considerar serán los siguientes;

1. Recolectar y ofrecer en tiempo real información sobre el estado del sistema y los diferentes dispositivos conectados al mismo.
2. Realizar la iteración principal en menor o igual tiempo que el requerido en la versión inicial.
3. La aplicación será monolítica.
4. La aplicación será desarrollada en Java.

El estudiante planteó la siguiente arquitectura para la prueba de concepto que permita cumplir los requisitos anteriores (Fig. 2.4).

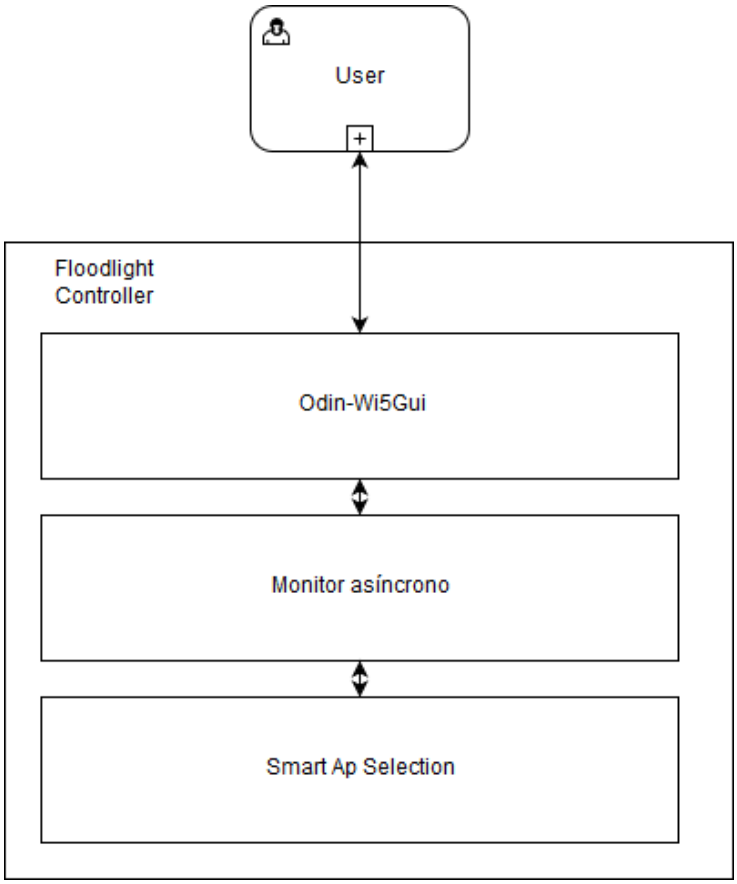

Figura 2.4, Arquitectura monolítica.

El desarrollo de la prueba de concepto se llevó a cabo durante dos semanas al comienzo del trabajo, cumpliendo todos los requisitos planteados anteriormente. Por tanto, el desarrollador llega a la conclusión de que el proyecto, tal y como se ha planteado en esta prueba de concepto, se puede llevar a cabo, teniendo en cuenta ciertos riesgos que se analizarán más adelante.

2.3.4 Investigación del sistema

De manera paralela al diseño y desarrollo de la prueba de concepto, se ha llevado a cabo un estudio sobre la totalidad del sistema sobre el que se va a trabajar.

El controlador sobre el que trabaja la aplicación *Smart AP Selection* se compone de múltiples servicios de bajo y alto nivel que conjuntamente forman el sistema *Odin-Wi-5*. Por tanto, para alcanzar una solución viable, se ha llevado a cabo un análisis de las diferentes capas que sostienen el proyecto (ver Fig. 2.5).

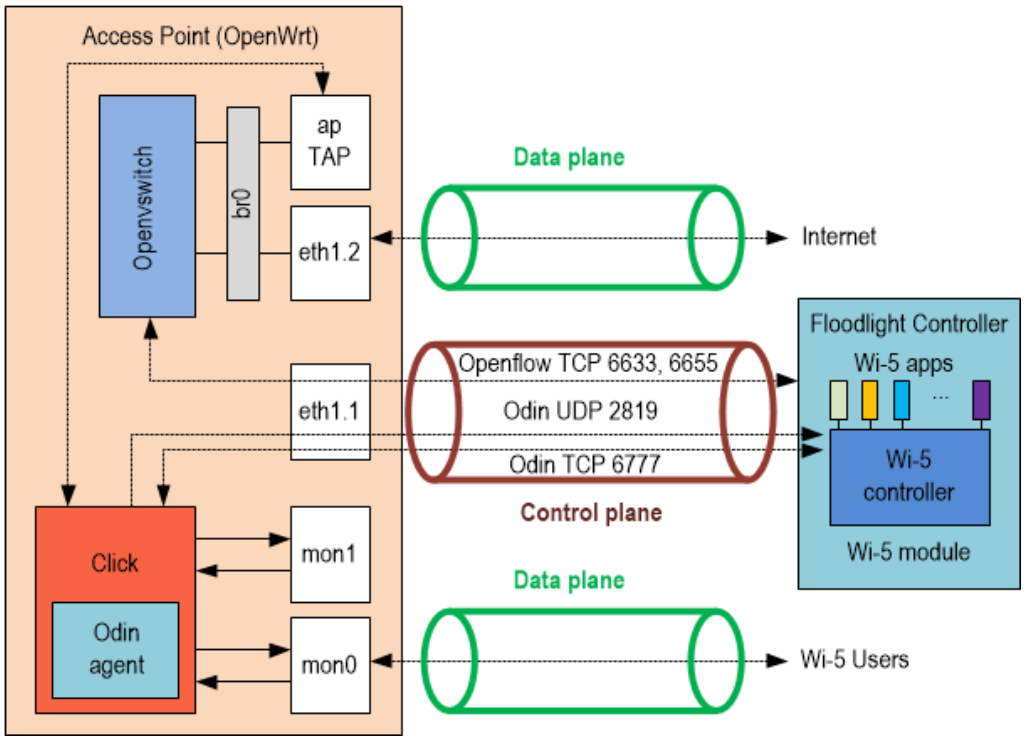

Fig 2.5. Elementos del sistema *Odin-Wi-5*

Como se indicó anteriormente (ver 2.1), la aplicación *Smart AP Selection* forma parte del Wi-5 controller, módulo iniciado por Lalith Suresh [15]. El módulo se encuentra a su vez en un controlador *OpenFlow* llamado *Floodlight Controller.* Se trata de un proyecto de gran envergadura, desarrollado a partir de *OpenFlow*, que no solo dispone de un controlador *OpenFlow*, sino que alrededor de dicho controlador fueron construidas un conjunto de aplicaciones que permitieran resolver futuras necesidades del usuario.

Floodlight Core dispone de múltiples aplicaciones y módulos. Tras el estudio de éste, se ha decidido destacar los siguientes:

- Servicio de almacenamiento *(Memory Storage Source)* con tipología NoSQL en memoria, que permite almacenar datos mientras el controlador esté en ejecución.

  El servicio posee ciertas limitaciones que impiden que sea utilizado como sistema de persistencia fiable: los datos se pierden en caso de que el sistema se apague, debido a que se almacenan en memoria. Por otra parte, el servicio no posee medidas para garantizar el aislamiento de los datos: en caso de que un módulo genere una tabla, otro módulo podría modificar libremente los datos de dicha tabla.

  Finalmente, este servicio sí posee medidas para evitar problemas de concurrencia y poder ser atacado desde todos los servicios presentes en *Floodlight*.

- Servicio *Api Server* (*Rest Api Server*). Servidor *Rest Api* dispuesto por el controlador *Floodlight* usando la librería de *RestletRoutable*. Este servicio puede ser usado por cada uno de los servicios presentes en *Floodlight*. Cada servicio puede crear sus propias peticiones *Rest Api*.

En caso de que el proyecto no fuera monolítico, hacer uso de este servicio podría permitir generar un servicio web donde se dispusieran los datos de la aplicación *Smart AP Selection*.

Esta información, junto a la prueba de concepto desarrollada en 2.3.1, han llevado al estudiante a realizar cambios significativos en los requisitos propuestos anteriormente (ver Sección 2.5).

2.4 Riesgos

El prototipo llevado a cabo en el apartado 2.3.3 ha dado como resultado el desarrollo de una aplicación monolítica que cumple los requisitos funcionales presentados en el apartado 2.3.3, y que por tanto debería ser capaz de cumplir el resto de los requisitos propuestos y alcanzar así una solución válida.

En el inicio del trabajo se conocían posibles riesgos que podrían ocasionar problemas a lo largo del proyecto. Durante el desarrollo de la prueba de concepto se han encontrado nuevos riesgos que se listan a continuación:

- **Rendimiento:** el desarrollo de la interfaz gráfica podría afectar al rendimiento de toda la aplicación, pudiendo llegar a incumplir el requisito técnico de "El tiempo de la iteración principal no debería empeorar".
- **Accesibilidad**: la aplicación se ejecutará únicamente donde se ejecute el sistema *Floodlight*, impidiendo acceder a ella desde otra máquina.
- **Inestabilidad**: El mal funcionamiento de la interfaz gráfica podría afectar al correcto funcionamiento del proceso que itera de manera continua en el servicio.
- **Mantenimiento**: El código generado, aunque dividido en capas, resulta más difícil de mantener para futuros desarrollos.
- **Concurrencia**: Los datos que genera la aplicación *Smart AP Selection* deben de ser transmitidos a otros servicios, por tanto, debe tenerse en cuenta los posibles problemas de concurrencia al atacar varios servicios los mismos datos.

2.5 Conclusión del análisis y requisitos finales

Con los resultados de la prueba de concepto y el estudio del sistema realizado, se plantea el siguiente dilema: ¿es posible llevar a cabo la solución del proyecto mediante la arquitectura y diseño presentados en la prueba de concepto? Sí, sin embargo, el desarrollador, en este caso el estudiante, considera que ligeros cambios sobre los requisitos técnicos permitirían desarrollar una solución mucho más robusta y mantenible.

2.5.1 Segunda captura de requisitos

Los cambios propuestos para los requisitos técnicos son los siguientes:

- En vista de que existe un servicio implementado en *Floodlight* que dispone de un servidor *ApiRest*, se propone sustituir la arquitectura monolítica por una arquitectura cliente-servidor.

- Desarrollar la interfaz gráfica como un cliente ligero en *Java* que se conecte al servidor HTTP, manteniendo el requisito funcional número 8.
- Hacer uso del servicio *MemoryStorageSource* de que dispone Floodlight, para crear una pasarela asíncrona entre el servicio web y la aplicación *Smart AP Selection*, eliminando el monitor y aprovechando los recursos del sistema.

Estos cambios mitigarían todos los riesgos asociados con anterioridad a una arquitectura monolítica, sin modificar los requisitos funcionales asociados al proyecto.

El cliente considera oportunos estos cambios y los acepta, modificando ligeramente con ello el requisito número 8.

2.5.2 Requisitos funcionales finales

Así pues, la lista definitiva de requisitos funcionales es la siguiente:

1. Se mostrará el estado de la red en tiempo real, incluyendo los puntos de acceso, estaciones conectadas y su estado de conexión.
2. Se podrá solicitar el *log* del sistema.
3. Se podrá visualizar una matriz de la atenuación medida entre las estaciones conectadas y los puntos de acceso.
4. Se podrá modificar algunos de los parámetros de la aplicación *Smart AP Selection*.
5. Se mostrará información referente a cada estación y punto de acceso.
6. Se podrá solicitar el escaneo de un canal por parte de cada punto de acceso.
7. Se podrá solicitar cambiar el canal de un punto de acceso.
8. Se podrá solicitar manualmente el *handoff* de una estación.
9. Se podrá solicitar información de la potencia escuchada de cada cliente.
10. Se podrá visualizar el tiempo en el aire, el número de paquetes y la potencia media de cada cliente mediante unas gráficas representativas.

Y esta es la lista de requisitos no funcionales:

1. La implementación de la solución se realizará utilizando la Java versión 8.
2. El grafo que dibuje la red se creará usando la librería GraphStream versión 1.3.
3. El sistema será robusto y mantenible a largo plazo.
4. El proyecto dispondrá de un cliente ligero conectado a un servidor *Web* a través de peticiones *HTTP*.
5. Se expondrán los datos de la aplicación *Smart AP Selection* mediante una *Api*.
6. El cliente debe ser multiplataforma.
7. El servicio web se comunicará con la aplicación *Smart AP Selection* mediante una pasarela asíncrona.

Capítulo 3 - Diseño

A la hora de realizar el diseño del sistema de acuerdo con los requisitos finales (Sección 2.5) y objetivos (Sección 1.2) del proyecto, se ha optado por un sistema cliente-servidor, permitiendo al servidor una mayor independencia y robustez frente a sistemas monolíticos. En este capítulo se abordan el diseño del sistema, los posibles cambios sobre la aplicación *Smart AP Selection*, la arquitectura y el diseño del cliente.

3.1 Arquitectura del sistema

En esta sección se aporta una visión global del sistema, mostrando su arquitectura (Fig 3.1) y un diagrama donde se representa el comportamiento de los diferentes elementos (Fig 3.2). Como se aprecia, la arquitectura del sistema está compuesta por diferentes capas:

- Una capa denominada *Odin*, que contiene la aplicación *Smart AP Selection* y los diferentes algoritmos que posee. En esta capa se realiza el intercambio de flujos de información entre el controlador y los diferentes AP.
- Una capa denominada *servidor*, que contiene los servicios intermediarios entre el cliente y la capa *Odin*, en este caso el servidor web y el servicio de almacenamiento NoSql.
- Y por último capa *cliente,* independiente del servidor, que ataca la capa *servidor* mediante peticiones HTTP.

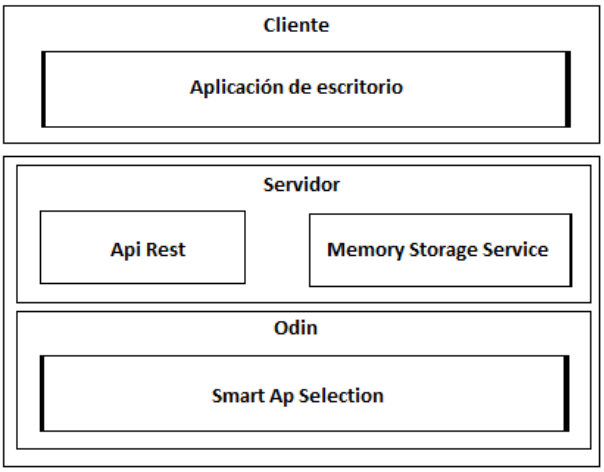

Figura 3.1. Arquitectura del sistema desarrollado

Por otra parte, la visión global del sistema (Fig 3.2) refleja el flujo que sigue la información, desde que es generada por los puntos de acceso, hasta que es consumida por el usuario final. Se ha evitado mostrar en el diagrama el flujo completo de la información a través del sistema Odin, y sólo se han mostrado los elementos que conciernen a este proyecto. Todos los elementos del diagrama serán explicados a continuación.

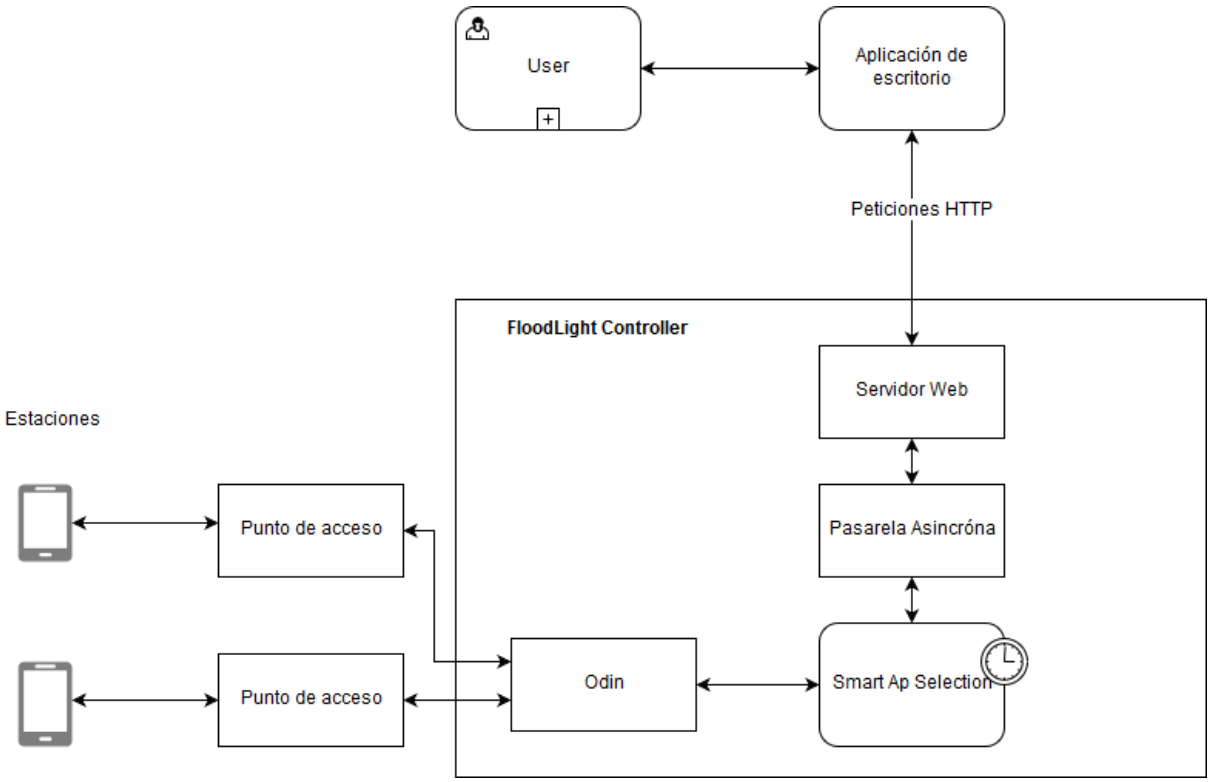

Figura 3.2. Visión global del sistema

3.2 Smart AP Selection

El flujo de la información (Fig 3.3) en la aplicación *Smart AP Selection* pasa por varios procesos diferenciados. La información que genera cada uno de ellos, así como la información obtenida directamente de los puntos de acceso, se vuelca sobre la pasarela, donde en caso de que algún cliente requiera de estos datos, se le enviarán los últimos actualizados.

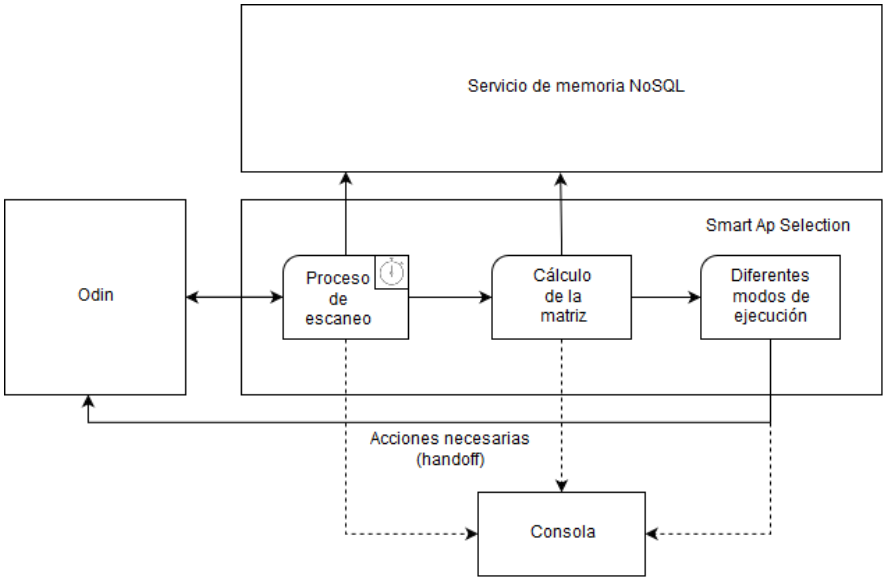

Figura 3.3. Flujo de datos en la aplicación *Smart AP Selection*

Antes de empezar la ejecución, este proceso realiza una pequeña etapa de inicialización, donde prepara las variables locales a partir de un conjunto de parámetros indicados por el usuario en un fichero de configuración. Estos son los parámetros que el usuario puede indicar antes de la ejecución de la aplicación:

- ***TimeToStart*** **(medido en segundos)**: Periodo inicial de tiempo en el cual la aplicación espera a que se inicialicen los puntos de acceso. Este parámetro ha resultado ser una limitación del sistema: si el tiempo se consume y alguno de los puntos de acceso no ha sido configurado, el sistema se bloquea, necesitando reiniciarse por completo. Por tanto, se propone que el usuario indique a la aplicación cuándo están listos el resto de los componentes mediante una ventana emergente.

- ***ScanningInterval*** **(medido en milisegundos)**: Los puntos de acceso deben escanear el medio durante un periodo de tiempo. Según cuanto tiempo dedique el punto de acceso al escaneo, obtendrá un número de datos variables: a mayor tiempo, una mayor cantidad de datos y viceversa. Este periodo se puede indicar como parámetro de la aplicación.

- ***AddedTime*** **(medido en milisegundos)**: Intervalo de tiempo añadido durante el cual la aplicación *Smart AP Selection* no solicitará más escaneos. Con esta pausa se permite a los APs realizar el resto de las tareas que tengan programadas, ya que los escaneos exigen un uso exclusivo de la interfaz de red.

- ***SignalThreshold*** **(medido en decibelio-milivatios, dBm)**: Parámetro usado en todos los modos. Representa el mínimo valor que puede alcanzar la atenuación de una estación antes de ser movida a otro punto de acceso.

- ***Hystheresis* (medido en segundos)**: Tras un *handoff* de una estación, este es el parámetro que indica el tiempo que debe esperar *Smart AP Selection* antes de realizar un nuevo *handoff* de la misma estación. De esta manera se evita el efecto "ping pong" (que una estación realice varios cambios seguidos entre dos APs).

- ***Alpha* (α, valor comprendido entre 0 y 1)**: El RSSI obtenido a través de los escaneos puede sufrir grandes variaciones debido a interacciones con el entorno por parte de la estación: esta pasa por detrás de una puerta, una persona cruza el espacio entre la estación y el punto de acceso, etc. Este tipo de perturbaciones hacen que el RSSI pueda variar de manera brusca. La aplicación *Smart AP Selection* "suaviza" esos resultados con el uso de escaneos históricos (ver la fórmula más abajo). El parámetro α indica el grado de suavizado, y las pruebas empíricas realizadas durante el proyecto Wi-5 indicaron que 0.8 es el valor recomendado.

- ***Pause* (medido en segundos):** Intervalo de tiempo entre las solicitudes de *handoff*, permitiendo al controlador realizar otras tareas.

- ***Mode* (Anexo 2)**: Indica el algoritmo que se ejecutará tras la creación de la matriz. Puede ser RSSI, FF, BALANCER, DETECTOR.

- ***Filename***: Nombre del fichero donde se aloja el *log* de la aplicación.

A continuación del escaneo periódico, el proceso calcula la matriz de atenuación, actualizando la información local de *Smart AP Selection*. Para cada par AP-estación, se realiza un cálculo en el cual se obtiene la atenuación suavizada (RSSI):

$$\text{RSSI suavizada} = \alpha * \text{nuevo RSSI} + (1- \alpha) * \text{RSSI histórico}$$

Una vez finalizado el proceso que calcula la matriz de atenuación, da comienzo el proceso que asigna las estaciones a sus puntos de acceso óptimos, teniendo en cuenta el RSSI suavizado y el número de estaciones que hay en cada AP.

Todo este proceso que realiza la aplicación *Smart AP Selection* debe cumplir un tiempo establecido por el requerimiento. Por tanto, cuando se afronte su refactorización debe tenerse tenerse en cuenta esta limitación impuesta por el proyecto.

Por tanto, para la refactorización de la aplicación se decide encapsular cada uno de los procesos identificados en la aplicación, eliminar todo el código no necesario y optimizar el restante.

En el diseño inicial, los tres procesos que componen la aplicación *Smart AP Selection* volcaban de manera continua información sobre la consola de Odin-Wi-5, siendo ésta la única manera de comunicarse con el usuario. Esta comunicación se realizaba en el mismo código de la función, pudiendo ocasionar problemas de rendimiento. Se ha decidido eliminar este código, aumentando así el rendimiento del proceso y reduciendo las líneas de código innecesarias.

3.3 Pasarela de comunicación asíncrona

Para el traspaso seguro de información entre la aplicación *web* y *Smart AP Selection* se cuenta con el servicio de memoria presente en el sistema *Floodlight* (Fig 3.2). Este servicio actúa como una base de datos *NoSql,* dando persistencia a los datos mientras el sistema esté en ejecución. Su estilo *NoSq*l requiere de la creación de entidades (ver Fig 3.4) donde se guarde la información que interesa al usuario. Dichas entidades, al generarse en memoria cacheada, deben ser creadas en cada ejecución del programa. Las entidades creadas se almacenarán en forma de tablas *NoSql* de objetos *Java*.

Las entidades creadas son las siguientes:

- Estaciones (*STA* en la Figura 3.4). Esta tabla contendrá toda la información concerniente a las estaciones, que se ha considerado relevante.
- Puntos de acceso (*AccessPoint* en la Figura 3.4). Los puntos de acceso poseerán su propia tabla donde se incluirá la información que es necesaria para cumplir los requisitos finales (Sección 2.5.2)
- Parámetros de la aplicación (*SmartApParameters* en la Figura 3.4). El requerimiento final 4 (Sección 2.5.2) requiere que algunos de los parámetros de la aplicación puedan ser modificados mientras la aplicación está en ejecución. Por tanto, será necesario que el usuario pueda cambiarlos y hacerlos llegar a la aplicación *Smart AP Selection*.

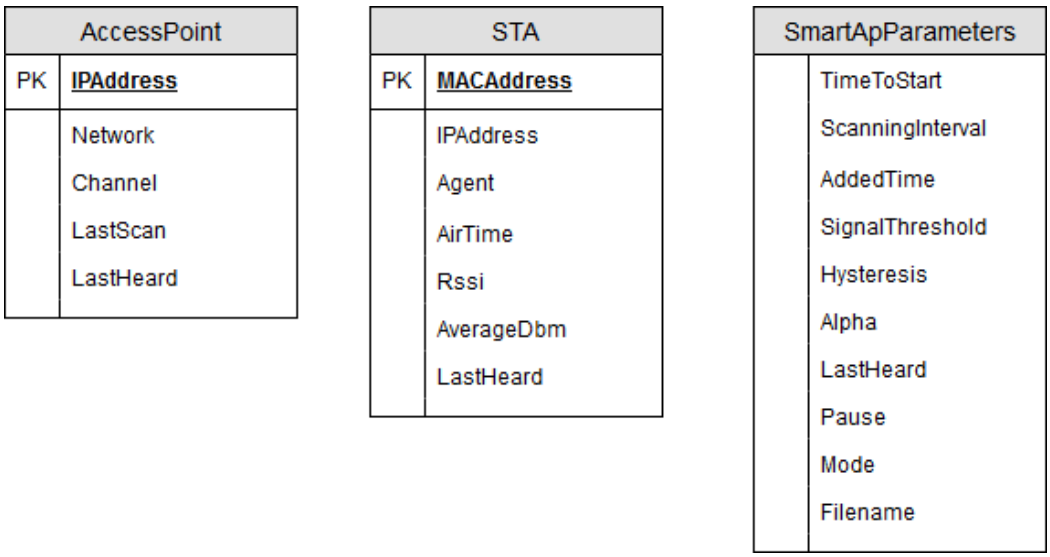

Figura 3.4. Tablas NoSQL presentes en la pasarela.

Aunque este servicio no posee las funcionalidades típicas de una base de datos, se ha decidido incluir una clave primaria en las dos primeras tablas. Estas claves servirán para identificar las diferentes entidades en el sistema y diferenciarlas. Al implementarse las funciones que inserten, borren o modifiquen los datos de la pasarela, se tendrán en cuenta estas claves evitando repetirlas. Las estaciones pueden variar de dirección IP, ya que se encuentran en un sistema de asignación dinámica de direcciones. Por tanto, se opta por utilizar el parámetro MACAddress como clave única. Los puntos de acceso poseen

direcciones IPv4 estáticas que usan para comunicarse con el controlador, siendo estas únicas y por tanto es seguro usarlas como clave primaria.

La entidad *SmartApParameters* refleja los parámetros que el usuario puede modificar durante la ejecución de la aplicación (requisito número 4), mencionados anteriormente (Sección 3.2).

A continuación, se detallan las entidades "*AccessPoint*" y "*STA*":

AcsessPoint:

- *IpAddress*: Dirección IPv4 usada para comunicarse a través de la red de control.
- *Network*: Nombre que identifica a la red Wi-Fi.
- *Channel* (intervalo entre 1 y 14): Canal Wi-Fi donde está funcionando el punto de acceso.
- *LastScan* (marca temporal): TimeStamp de la última vez que el punto de acceso realizo un escaneo solicitado por la aplicación *Smart Ap Selection*.
- *LastHeard* (marca temporal): TimeStamp de la última vez el controlador recibió alguna clase de mensaje del punto de acceso.

STA:

- *MACAddress*: Dirección MAC (Media Access Control, control de acceso al medio) física de un dispositivo. Se trata de una dirección única e irremplazable.
- *IpAddress*: Dirección IPv4 asignada a la estación por el DHCP.
- *Agent*: Dirección IPv4 que indica a qué punto de acceso está conectado esta estación actualmente.
- *Airtime* (Medido en milisegundos): Tiempo de aire que ha ocupado una estación durante el último escaneo. Se obtiene como la suma de los tiempos que han consumido las tramas Wi-Fi generadas y recibidas por esa estación.
- *Rssi* (Medido en decibelios-milivatio): RSSI sin suavizar obtenido en el último escaneo solicitado.
- *AverageDBM* (Array de números decimales): Este *array* de datos contiene la atenuación suavizada de una estación con cada uno de los puntos acceso. Este objeto único de cada estación, en conjunto con el resto de las estaciones, conforma la matriz de atenuación.
- *LastHeard* (marca temporal): TimeStamp de la última vez que un punto de acceso identificó la estación en la red.

Para cada una de las tablas se prevé una cantidad de tráfico diferente. La tabla donde se almacenarán de manera local las estaciones y sus atributos tendrá una mayor cantidad de modificaciones, debido a que, con cada iteración del proceso, estos datos se modifican. Las entidades que representan los puntos de acceso y los parámetros de la aplicación recibirán un tráfico mucho menor, ya que se usan principalmente para ejecutar las peticiones del usuario, por ejemplo, el cambio de canal o el cambio de un parámetro.

A lo largo de la ejecución del proceso, si se deja de escuchar una estación, el sistema es incapaz de diferenciar entre si ésta ha salido de la red o se ha apagado. Por tanto, ha de

considerarse el atributo *LastHeard* como el límite de vida de una estación. El sistema tiene en cuenta que, si no ha escuchado a una estación en un período de tiempo, los datos de dicha estación no se mostrarán cuando se solicite información por las estaciones conectadas, pero seguirán almacenados en el servicio.

3.4 Servicio web

El servicio web, juntamente con la pasarela (Sección 3.3), actúa de intermediario entre el cliente y la aplicación *Smart AP Selection* independizando un servicio del otro.

Dicho servicio web dispondrá de los siguientes endpoints:

1. Se plantea un endpoint donde se obtengan todos los clientes que están actualmente conectados o lo hayan estado en algún momento.
2. Se plantea un endpoint donde se obtengan todas las estaciones que estén actualmente conectadas.
3. Se plantea un endpoint donde se obtengan todos los agentes que actualmente estén conectados.
4. Se plantea un endpoint donde se indique al sistema que un agente debe cambiar el canal donde está emitiendo. Se debe indicar la dirección IPv4 del agente y el canal al cual se desea cambiar.
5. Será necesario un endpoint donde se indique la solicitud del *handoff* de una STA de manera manual. El *handoff* requerirá de dos parámetros: la dirección IP del agente al que se desea mover la estación, y la dirección física MAC que identifica el cliente.
6. Se implementará una petición que permita que el usuario pueda solicitar un escaneo de un canal por parte del agente. Los agentes pueden escanear en cualquier momento, si no se da el caso que se encuentren ya escaneando. En ese caso, este endpoint rescata los datos del último escaneo e informa de ellos al usuario. Al estar escaneando el entorno de manera periódica, estos datos se encontrarán desfasados por escasos milisegundos, por lo que se informará al usuario de que dichos datos han sido obtenidos de escaneos anteriores.
7. Se han creado dos endpoints con relación a los parámetros de la aplicación: una petición para obtener los parámetros y otra para solicitar la modificación de alguno de ellos. Entre los parámetros de la aplicación hay dos que no se permitirá modificar al usuario: el *Modo* o algoritmo que se está ejecutando actualmente, y la dirección y nombre del fichero *log* del sistema.
8. Para resolver el requerimiento que permite mostrar al usuario una matriz de atenuación entre las estaciones y los agentes, se diseña un endpoint que envíe dicha matriz.

Todos y cada uno de los endpoints propuestos responden mediante objetos *JSON.* De esta manera se estandariza la comunicación entre los servicios.

3.5 Diseño del cliente

El cliente se desarrollará como una aplicación de escritorio que mediante peticiones HTTP obtenga la información que requiere de la aplicación *Smart AP Selection.* Con el fin de realizar una capa mantenible y robusta, se opta por un modelo modelo-vista-controlador [16] (MVC) (Fig 3.5). Dicho modelo subdivide una aplicación en tres partes principales, con el objetivo de separar la lógica de negocio y sus datos, de la capa donde interactúa el usuario y todos los controladores encargados de responder a los eventos generados.

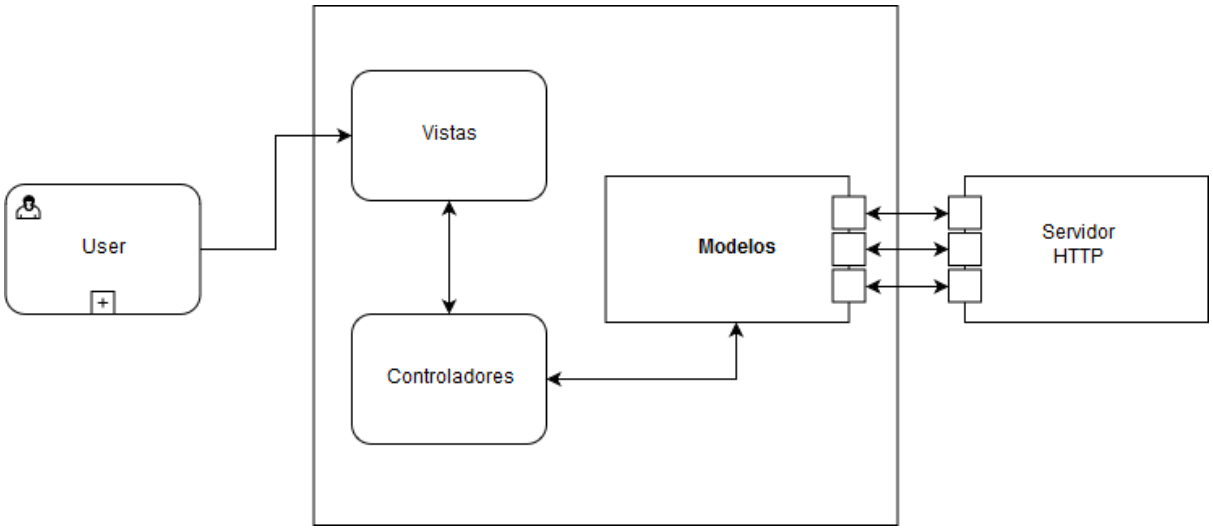

Figura 3.5. Modelo MVC de la aplicación de escritorio.

La capa de acceso a datos se compone de un conjunto de peticiones al servicio *HTTP* dispuesto por el controlador.

Se han diseñado las diferentes vistas de la aplicación de tal manera que cumplan los requisitos finales propuestos por el cliente, presentando los datos de la red de una manera amigable y en tiempo real.

Se diseñan cuatro vistas diferentes acordes a los diferentes casos de uso del usuario. Cada una tiene el objetivo de mostrar de una manera agradable y útil la compleja información que produce el sistema.

3.5.1 Pantalla general

La vista en tiempo real mostrará el estado de la red mediante la librería GraphStream, y se tomará de ejemplo a seguir la Fig. 3.4:

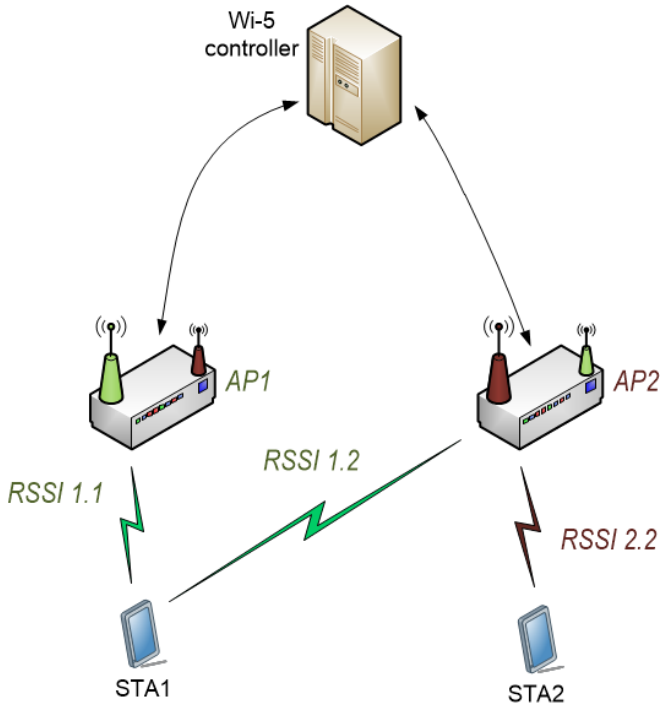

Figura 3.4. Elementos del sistema Odin-Wi-5

Se propone un gráfico piramidal, con el controlador arriba, los diferentes puntos de acceso un nivel por debajo, los nodos de las estaciones (STA) otro nivel por debajo, y representando las conexiones mediante líneas continuas entre los nodos. Las estaciones indicarán el RSSI en el gráfico, tal y como se mostraba en la aplicación inicial.

Se identificará a los componentes de la red por diferentes iconos, ya que la librería *GraphStream* permite modificar los nodos y aplicarles imágenes. A las estaciones se les asigna el icono de un móvil, ya que resultan ser las estaciones que más comúnmente se conectan a las redes Wi-Fi. El estado de la conexión de una estación se identificará con diferentes colores, en función de la calidad de esta:

- Verde, si el RSSI suavizado es menos de la mitad del parámetro *SignalThreshold*. Se usará el valor absoluto. Por ejemplo, si el threshold es 50 dBm, aparecerán en verde todas las estaciones cuyo RSSI sea igual o menor que 25 dBm.
- Amarillo, si el RSSI suavizado está entre el valor asignado al verde y el propio parámetro *SignalThreshold*.
- Rojo, para toda estación cuyo valor absoluto de señal supere el parámetro *SignalThreshold*.

La librería *GraphStream* tiene una limitación: no es posible configurar las posiciones de los diferentes nodos de un grafo respetando una jerarquía piramidal. Las posiciones de los nodos deben asignarse de manera absoluta. Se decide, por tanto, usar una librería de Java que permita crear un diseño en árbol, que actúe de esqueleto del grafo. La librería escogida es Abego Treelayout Core [17], que permite crear árboles de nodos y asignar una separación entre las diferentes hojas del árbol, permitiendo de una manera simple obtener las

coordenadas absolutas de cada nodo y pintarlos sobre el gráfico, formando una jerarquía piramidal.

Juntamente con este gráfico, se planea añadir los parámetros de la aplicación. En esta vista el usuario puede realizar los siguientes casos de uso:

- El usuario podrá ver el estado de la red en tiempo real, incluyendo los puntos de acceso, estaciones conectadas y su estado de conexión.
- El usuario podrá solicitar el log del sistema.
- El usuario podrá visualizar una matriz de atenuación entre las estaciones conectadas y los puntos de acceso.
- El usuario podrá modificar algunos de los parámetros de la aplicación *Smart AP Selection*.

3.5.2 Pantalla para los puntos de acceso

En la vista perteneciente a los puntos de acceso se plantea mostrar la información referente a los puntos de acceso y las diferentes acciones que se pueden realizar sobre estos:

- El usuario podrá ver información referente a cada estación:
    - Dirección IPv4 actual.
    - Canal donde está actualmente la estación
    - Identificador de la red a la que está conectada la estación.
    - El número de clientes conectados.
    - La última vez que la estación realizó un escaneo.
    - La última vez que se escuchó a la estación.
- El usuario podrá solicitar el escaneo de un canal por parte de cada punto de acceso.
- El usuario podrá solicitar cambiar el canal de un punto de acceso.

3.5.3 Pantalla para las estaciones

La vista perteneciente a las estaciones se plantea igual que la anterior, pero exclusivamente para las estaciones, siendo éstas las acciones que puede llevar a cabo un usuario:

- El usuario podrá ver información referente a cada estación:
    - Dirección IP asignada por la red.
    - Dirección física MAC de la estación.
    - A qué agente está conectada actualmente.
    - La última vez que se escuchó a esta estación.
- El usuario podrá solicitar manualmente el *handoff* de una estación.
- Se podrá solicitar una matriz de atenuación únicamente del cliente y los puntos de acceso.

3.5.4 Pantalla para las gráficas

Por último, la vista perteneciente a las gráficas del sistema presentará de manera gráfica los datos del último escaneo. Estas son las gráficas que se han considerado útiles:

- El tiempo en el aire de todas las estaciones conectadas a un punto de acceso escogido.
- El RSSI suavizado de cada estación para un punto de acceso escogido.
- El número de paquetes de cada estación para un punto de acceso escogido.

Estos tres parámetros permiten al administrador de la red comprobar que el algoritmo de balanceo está funcionando correctamente. Si una estación ocupa mucho tiempo en el aire, enviando pocos paquetes y con un RSSI bajo, esta estación estará probablemente posicionada en un punto de acceso no óptimo, recibiendo peor señal de la que podría estar recibiendo. En el caso opuesto, una estación con buena señal debería ser capaz de enviar grandes cantidades de paquetes ocupando un menor tiempo de aire.

Capítulo 4 - Desarrollo e implementación

Una vez finalizada la fase de diseño (Capítulo 3) se procede a realizar la implementación de los diferentes componentes. Como se ha explicado con anterioridad, la implementación del sistema se realiza enfocando la solución al caso más relevante, sirviendo esta solución como base para futuros proyectos.

Esta implementación se realiza en dos partes diferenciadas, afrontando cada una de ellas de manera independiente. En una primera fase se realizan los desarrollos sobre el controlador o servidor, implementando el servicio web, la pasarela y la refactorización de la aplicación *Smart AP Selection.* Y la segunda fase consiste en el desarrollo de la aplicación de escritorio atacando el ya establecido servicio web. Durante cada uno de los desarrollos se realizan las pruebas que se consideran necesarias para asegurar la calidad de la solución final.

4.1 Servidor

Esta sección explica los pasos dados para la correcta implementación de todos los desarrollos diseñados en el capítulo anterior que tengan alguna relación con el controlador externo al cliente, como puede ser la implementación del servidor web o la pasarela asíncrona *NoSql*.

4.1.1 Problemas generales encontrados

A la hora de desarrollar sobre el controlador Odin-Wi-5, el estudiante ha encontrado algunos problemas generales a la hora de trabajar con el sistema.

Los puntos de acceso cedidos por el grupo de investigación para las pruebas y desarrollo de este Trabajo Fin de Grado se han modificado con un módulo alterado de *Click Modular Router* [18], cambiando su comportamiento. Está modificación implica que algunas estaciones no sean capaces de conectarse correctamente a la red.

A la hora de afrontar este problema el estudiante decidió preparar un ordenador de sobremesa, que conocía de antemano que era capaz de asociarse a la red, con una tarjeta de red conectada a un cable USB de 25 metros (Fig 3.5), permitiendo al desarrollador “mover” la estación a lo largo del pasillo de la segunda planta del edificio del Ada Byron, realizando así las pruebas necesarias y usando esta estación como cliente principal del sistema.

Figura 3.5. Foto de la tarjeta de red conectada al cable usb de 25 metros.

Alcanzada la mitad del desarrollo del servidor, se detecta que las estaciones se consiguen asociar a la red, pero el router DHCP no les asigna ninguna dirección IP. El estudiante, al capturar el tráfico entre el controlador y el DHCP mediante la herramienta *tcpdump,* descubre que el servidor DHCP no recibe en ningún momento los paquetes desde los puntos de acceso, ocasionando con ello que no sea capaz de asignar dirección IP a las estaciones.

La razón de ello era que de manera paralela el proyecto de investigación se estaba llevando a cabo desarrollos sobre otra parte del proyecto. Estos desarrollos se estaban realizando en lo que se creía que era una red aislada de la red del Trabajo Fin de Grado. Las dos redes eran completamente idénticas, mismos componentes, incluso mismas direcciones IPv4 estáticas.

Las dos redes resultaban no estar aisladas la una de la otra, lo que ocasionaba conflictos a la hora de recibir y redirigir los paquetes. Los paquetes con dirección del servidor DHCP de la red del desarrollador eran capturados por el servidor de la red adyacente, al tener la misma IP y funcionalidad, ocasionando múltiples problemas de conectividad entre los componentes.

Una vez separadas y aisladas las redes, se obtuvo una mejora de la estabilidad y el rendimiento del sistema Odin-Wi-5 en ambos sistemas, eliminando los problemas encontrados al intentar asociar y conectar una nueva estación a la red.

4.1.2 Implantación de los cambios en la aplicación *Smart AP Selection*

Se realiza una primera aproximación identificando las trazas de la aplicación, estas líneas de código ahora no se consideran útiles y por tanto se han eliminado. Durante esta aproximación se ha encontrado el siguiente patrón de código múltiples veces:

```
if(client_dBm[vip_index]>SMARTAP_PARAMS.signal_threshold) {
  cntx.lastAgentAddr = agentAddr;
  flowsReceived.put(clientAddr,cntx);
  handoffClientToAp(eth,vipAPAddr);
} else {
  System.out.print([Flow] Detected flow from client " + clientAddr \n");
}
```

Como se aprecia, el código encapsulado en "else" es meramente informativo. Abusar de esta clase de condicionales dificulta la legibilidad del código. Refactorizando estos pequeños trozos de código, se aumenta la legibilidad, reduciendo también el número de líneas.

La clase *SmartApSelection* posee un conjunto de métodos privados que realizan múltiples tareas, calculan variaciones, obtienen parámetros de mapas de atributos. Todos estos métodos son muy simples, pero se utilizan con el fin de reutilizar código. Estas pequeñas funciones se podrían considerar muy útiles en cualquier aplicación que interactúe con *Odin*. Por tanto, se decide hacerlas públicas y almacenarlas en una clase *Helper*, donde han sido comentadas y refactorizadas.

Iniciar *Smart Ap Selection* requiere de un conjunto de líneas de código muy simples y si se organizan correctamente permite a un futuro desarrollador dedicar muy poco esfuerzo añadir nuevas funcionalidades. El código que inicia la aplicación se ha dividido en: inicialización de los parámetros globales y inicialización de las entidades de la pasarela.

Tras estos cambios, se continúa encapsulando el código de los diferentes algoritmos presentes en la aplicación, esta encapsulación permite afrontar cada una de las partes del código de una manera mucho más simple y personalizada. Cada uno de los algoritmos añadía entre 15 y 20 líneas de código fácilmente encapsulables, y algunas de estas nuevas funciones reutilizaban de manera continua las mismas variables. Por ejemplo, al inicio de la aplicación se creaba una variable numérica que actuaba de índice, esta variable se reiniciaba a cero tras finalizar un conjunto de operaciones para ser nuevamente utilizada. Este mecanismo dificulta la comprensión del código a cambio de utilizar menos recursos.

En la parte final del código de la aplicación, se creó una pequeña función acompañada de una clase para detectar flujos, usado en el modo *DETECTOR*. Crear clases con diferentes funcionalidades dentro de otras clases va en contra de las ideas que transmite *SINGLE RESPONSABILITY*, primer principio de los principios *SOLID* para diseño de clases. Cada clase debe tener una finalidad sencilla y concreta, por tanto, incluir un conjunto de métodos para detectar flujos de información en la clase *Smart AP Selection* daría dos funcionalidades

a esta clase, negando el primer principio. Las funciones que trabajan con estos flujos se han independizado en una nueva clase con esa finalidad.

Durante la asignación de las estaciones a nuevos puntos de acceso, se puede calcular el RSSI óptimo mediante dos algoritmos: el algoritmo de balanceo y el algoritmo *Fittingnes Factor*. Mantener el código de ambos en la función principal suponía dificultar la comprensión del código, refactorizar ambos algoritmos ha reducido la posibilidad de futuros errores y mal funcionamiento.

Algunos de los bucles de la función recorrían todos los elementos buscando uno en particular y al encontrarlo no salían del bucle, sino que seguían iterando. Si los datos sobre los que iterara el bucle fueran masivos, el coste de ejecución de este código se incrementa exponencialmente.

```
boolean stasToMove = false
// If there is not a STA with enougth RSSI, not handoff
for (MACAddress eth: rssiData.keySet()) {
   Double [] client_dBm = rssiData.get(eth);
   if (client_dBm[ind_aux] > SMARTAP_PARAMS.signal_threshold) {
      stasToMove = true;
   }
}
if(stasToMove)
{
......
}
```

Como resultado, las 700 líneas que ocupaba la iteración principal se han reducido a 55, encapsulando el código y reduciendo al mínimo el número de trazas. A la hora de analizar o comprender el código, es mucho más sencillo ir recorriendo funciones documentadas con código encapsulado que una única interacción sobre un bucle, complejo y difícil.

Con estos pequeños cambios se ha conseguido reducir el coste del proceso en casi un milisegundo, aumentando con ello la legibilidad del código y la mantenibilidad de cara al futuro.

Ahora que muchas de las funcionalidades se han encapsulado, se pueden realizar pruebas unitarias por cada una de las funciones, validando así su correcto funcionamiento.

4.1.3 Implantación de la pasarela NoSQL

Como se explica en la fase de diseño (Sección 3.3), la pasarela de datos, diseñada a partir de un servicio de que implementa *FloodLight*, es la base de datos intermedia que alberga los datos para la aplicación web.

En el primer acercamiento a la implantación de la pasarela se desarrolla una función que inicia las tablas *NoSql* al inicio del proceso de la aplicación, antes de iniciar los escaneos periódicos. Las tablas se crean tal y como se han diseñado en el capítulo anterior y de manera paralela se desarrollan las diferentes funciones que atacarán estas tablas usando operaciones CRUD. Dado que el servicio puede almacenar objetos Java en las tablas, se generan objetos con los mismos parámetros que se diseñaron en el capítulo anterior.

En esta misma fase se realizan las modificaciones necesarias en el código de la aplicación *Smart AP Selection* para poder interactuar con el servicio que actúa de pasarela. Durante el período entre la finalización del proceso y el reinicio del escaneo periódico se comprueba si el cliente ha realizado alguna petición para modificar los parámetros de la aplicación y, si es así, se realiza la modificación antes de que el bucle principal se reinicie (Fig 3.6).

Las modificaciones sobre la pasarela se realizan a la vez que el sistema actualiza los parámetros locales para los diferentes cálculos posteriores.

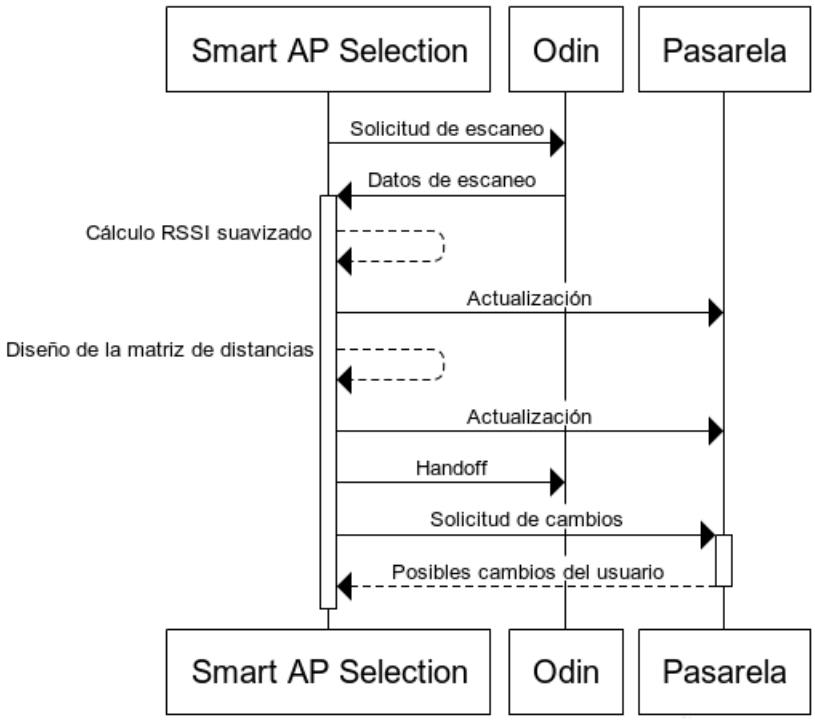

Figura 3.6. Diagrama de secuencia de la información

Tal y como se ha nombrado anteriormente, *Floodlight* posee un sistema interno de pruebas unitarias que se ha utilizado para desarrollar las pruebas que validan todas las funciones CRUD de la pasarela, así como la creación de las tablas y sus atributos.

4.1.4 Implantación de la API

Tras la implementación de la pasarela de datos, se procede a implementar la web Api tal y como se diseñó en el capítulo anterior. El trabajo se divide en tres partes, identificando cada

parte con uno de los componentes del sistema: aplicación general, puntos de acceso y estaciones.

Tras un primer acercamiento mediante el uso de la pasarela de datos, el estudiante encuentra que algunos de los datos son accesibles desde fuera de la interacción principal de la aplicación, como por ejemplo los parámetros de la aplicación o el log. Esto se debe a que la propia aplicación *Smart AP Selection* hereda algunas de las funciones de la clase *OdinMaster,* donde se inicializa el log y los parámetros de la aplicación.

Por tanto, el desarrollador propone una alternativa a realizar continuas llamadas a la pasarela en busca de posibles datos actualizados, y se considera que algunos de los datos que no van a ser actualizados, o se disponen de manera pública por la aplicación *OdinMaster* no sean modificados por la aplicación *Smart AP Selection.* De esta manera se reducen las comunicaciones entre la pasarela y la aplicación, actualizando sólo los datos que se consideren esenciales, como por ejemplo el RSSI suavizado o los cambios que solicite el usuario a través de la aplicación de escritorio.

Los diferentes *endpoints* propuestos en el capítulo de diseño se han realizado mediante la librería *RestletRoutable* presente en el sistema. Se ha generado una función por cada uno de los endpoints y se han agrupado en clases por datos atacados: una clase para los endpoints referentes a los agentes, otra para los endpoints referentes a los clientes y una última para los endpoints de la aplicación.

A falta de un cliente que consuma los endpoints, se ha optado por una aplicación de apoyo como es *Postman [19],* aplicación que permite realizar peticiones HTTP configuradas por el desarrollador, comprobando así el correcto funcionamiento de las funciones.

El sistema de pruebas unitarias presente en el sistema *Floodlight* ha permitido desarrollar pruebas en *Junit [20]* para cada uno de los diferentes endpoints garantizando la calidad de la Api.

4.2 Cliente

Una vez finalizadas todas las mejoras del controlador, el estudiante comienza los desarrollos de la aplicación de escritorio, tal y como se describen en el capítulo de diseño. Se da por supuesto que, si el desarrollador encuentra alguna alternativa mejor al diseño previsto, realizará las debidas modificaciones con el fin de alcanzar una solución más robusta y mantenible.

Debido a que el sistema proporcionado por la universidad no puede accederse desde fuera del laboratorio, el estudiante tuvo que desarrollar de manera independiente una Api basada en *Spring Boot [21]* que actuara como controlador local, reflejando datos estáticos obtenidos anteriormente de la aplicación principal en ejecución. De esta manera el estudiante es capaz de trabajar en el desarrollo del cliente fuera del laboratorio de la universidad.

A continuación, se muestran las diferentes pantallas desarrolladas que cumplen los requisitos funcionales finales, y las diferentes decisiones tomadas durante su desarrollo e implementación.

4.2.1 Vista general

La vista general necesita de datos actualizados de manera constante, por tanto, se desarrolla una tarea que, de manera periódica, solicita los datos del último escaneo, obteniendo los últimos cambios que se han producido en la red. Esta tarea se realiza con *Timeline,* componente de la librería de *JavaFX,* y se ejecuta cada tanto milisegundo como se indique en el parámetro *intervalo de escaneo* (parámetros de la aplicación). Una vez obtenidos los datos, se actualizan los nodos presentes en el grafo, pudiendo ocurrir las siguientes acciones:

- Ninguna estación ha cambiado y por tanto se actualizan los parámetros como el tiempo en el aire, el número de paquetes o el RSSI suavizado.
- Se ha producido un *handoff* en el sistema. Por tanto, se deben recalcular todas las posiciones de las estaciones mediante el algoritmo en árbol, para asignar nuevas posiciones absolutas a los nodos de acuerdo con el nuevo estado de la red, y redibujar el grafo.
- Una estación está alcanzando de manera progresiva el valor mínimo de *Threshold* indicado en los parámetros de la aplicación. En caso de que se produzca esto, se dibujarán nuevas conexiones de la estación con los demás puntos de acceso. De esta manera, el usuario puede apreciar en tiempo real los diferentes datos que posee la aplicación y las posibles futuras decisiones que tomará la aplicación en base a estos datos. Estas nuevas conexiones siempre serán amarillas y dibujadas con líneas discontinuas.

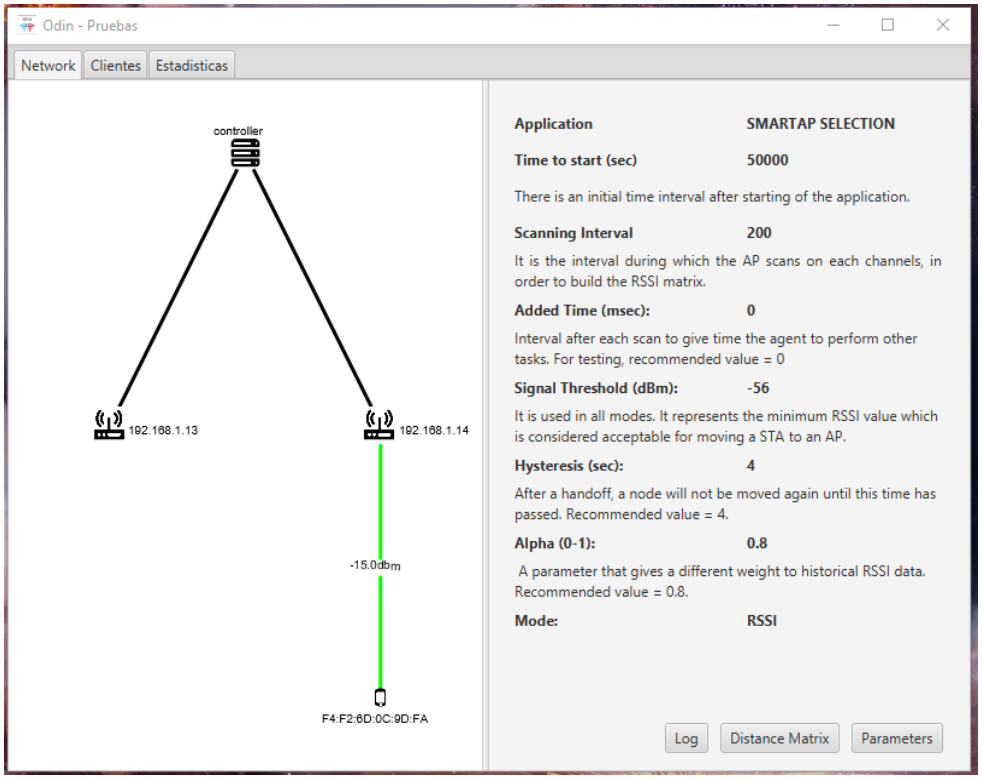

Figura 3.7. Captura de pantalla de la vista general.

Para la posible modificación de los parámetros, se ha incluido una pantalla emergente con información adicional para el usuario, como por ejemplo cuáles son los valores recomendados.

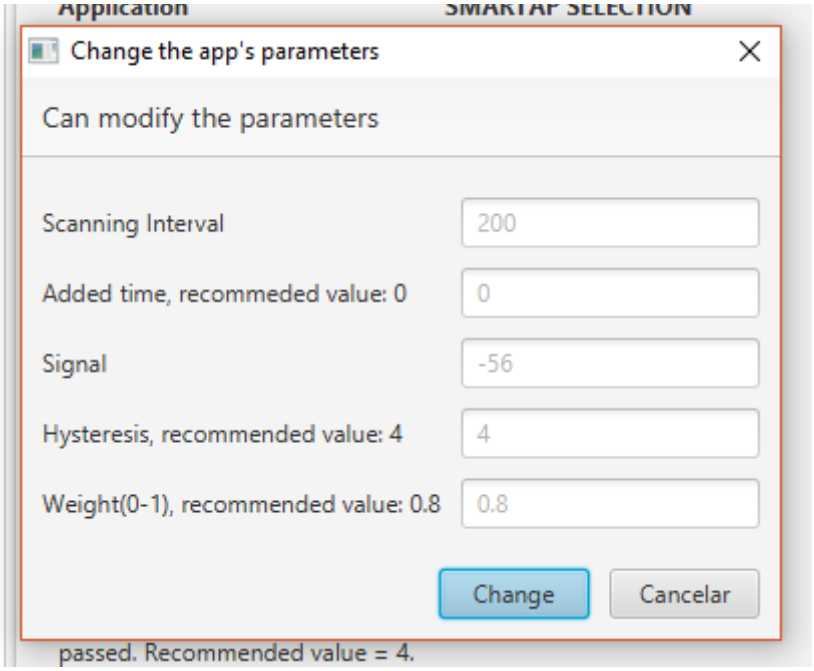

Figura 3.8. Captura de pantalla del pop up de parámetros

La matriz de distancias entre puntos de acceso y clientes se muestra con los mismos colores que se muestran en el gráfico inicial, indicando de esta manera al usuario la calidad de las diferentes conexiones entre puntos de acceso y estaciones.

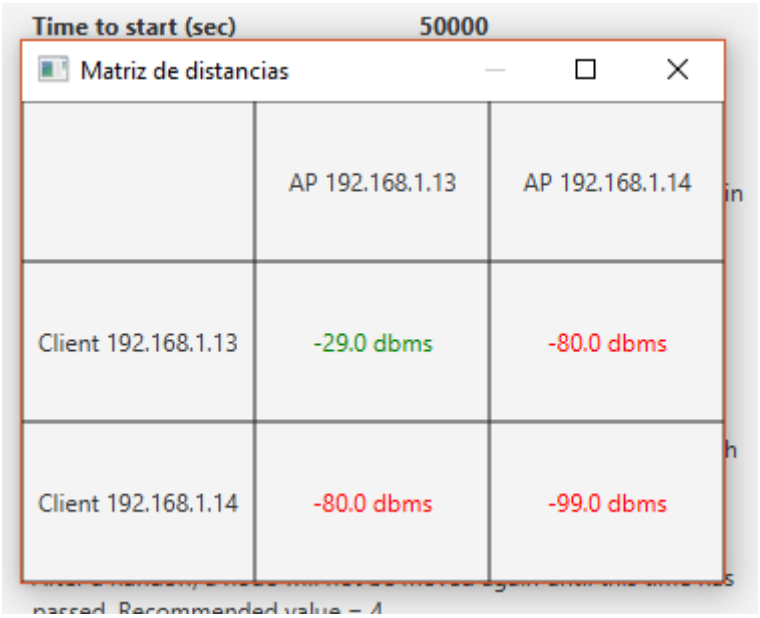

Figura 3.9. Captura de pantalla del pop up con la matriz de atenuación

4.2.2 Agentes y clientes en detalle

Durante el desarrollo de las dos vistas propuestas, una para los agentes y otra para los clientes, el grupo de investigación propone fusionar ambas en una única vista. Al compartir las estaciones y los agentes múltiples parámetros, se opta por una vista usando una tabla en árbol, incluyendo en la tabla tanto estaciones como puntos de acceso, diferenciándolos por una columna.

Durante los primeros acercamientos a esta pantalla, el desarrollador opta por mostrar la dirección IP de las estaciones, pero acaba por rechazarlo, debido a que los clientes no se pueden identificar por su IP en una red que asigna dichas direcciones de manera dinámica a través de un servidor DHCP, siendo esta una clave no única. Por tanto, se decide que en toda la aplicación los clientes se identificarán mediante la dirección MAC física.

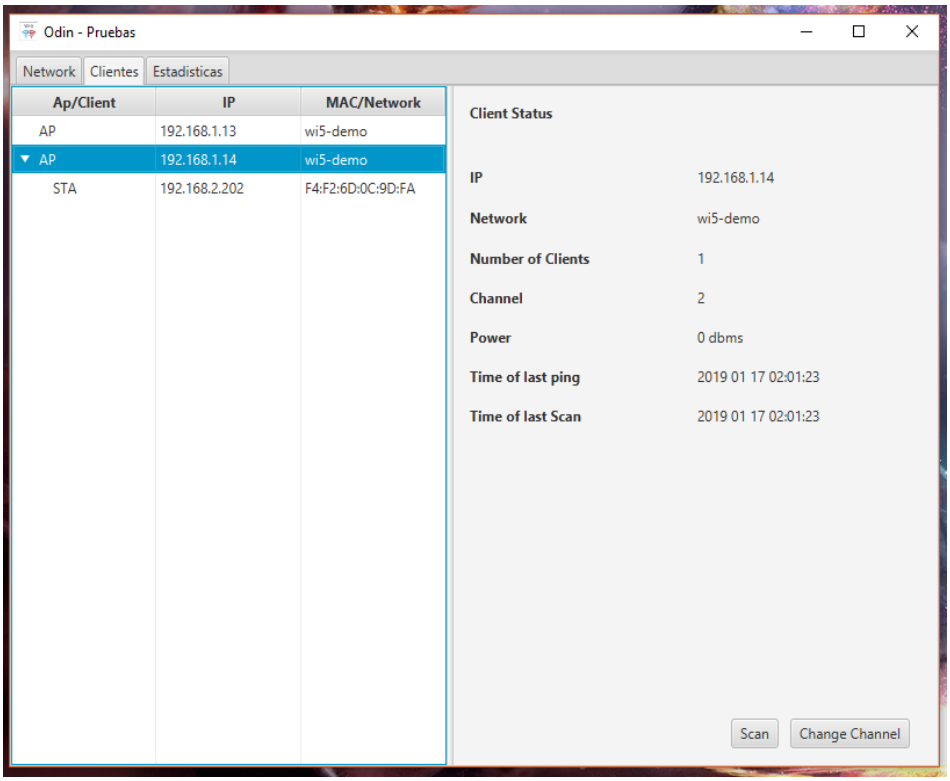

Figura 3.10 Vista *agentes*.

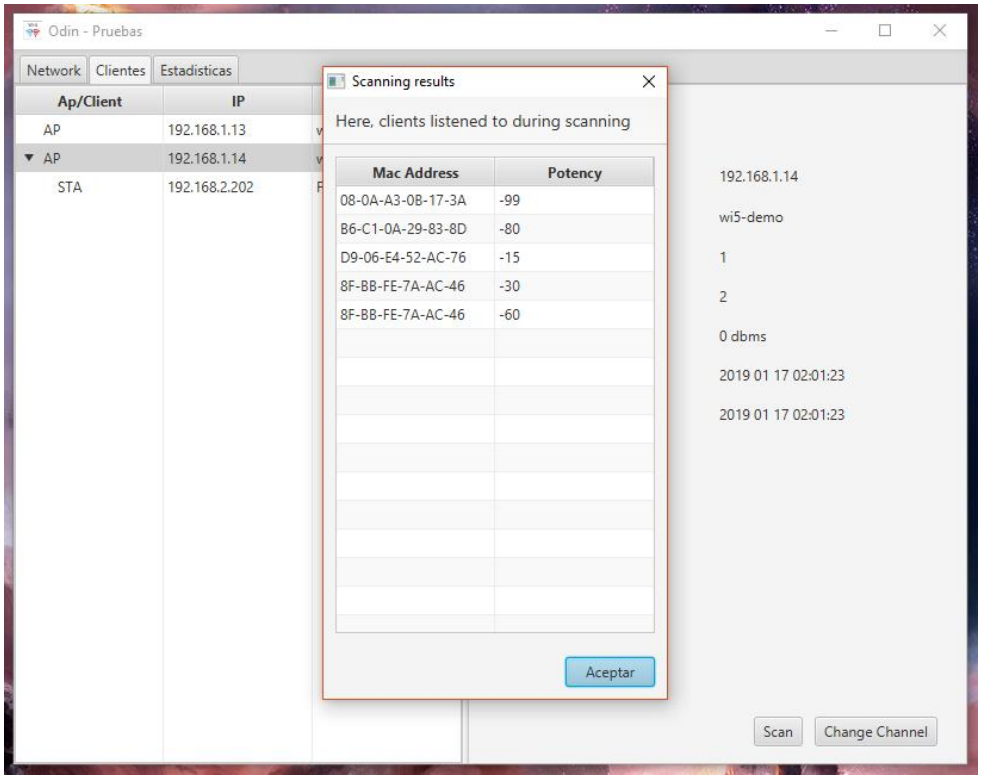

Figura 3.11. Ventana emergente *resultados escaneo*.

En caso de tener seleccionado un agente y pulsar uno de los dos botones inferiores, ambos mostrarán dos ventanas emergentes, que se muestran a continuación, identificando cada una de ellas con uno de los requisitos finales propuestos en el Capítulo 2.

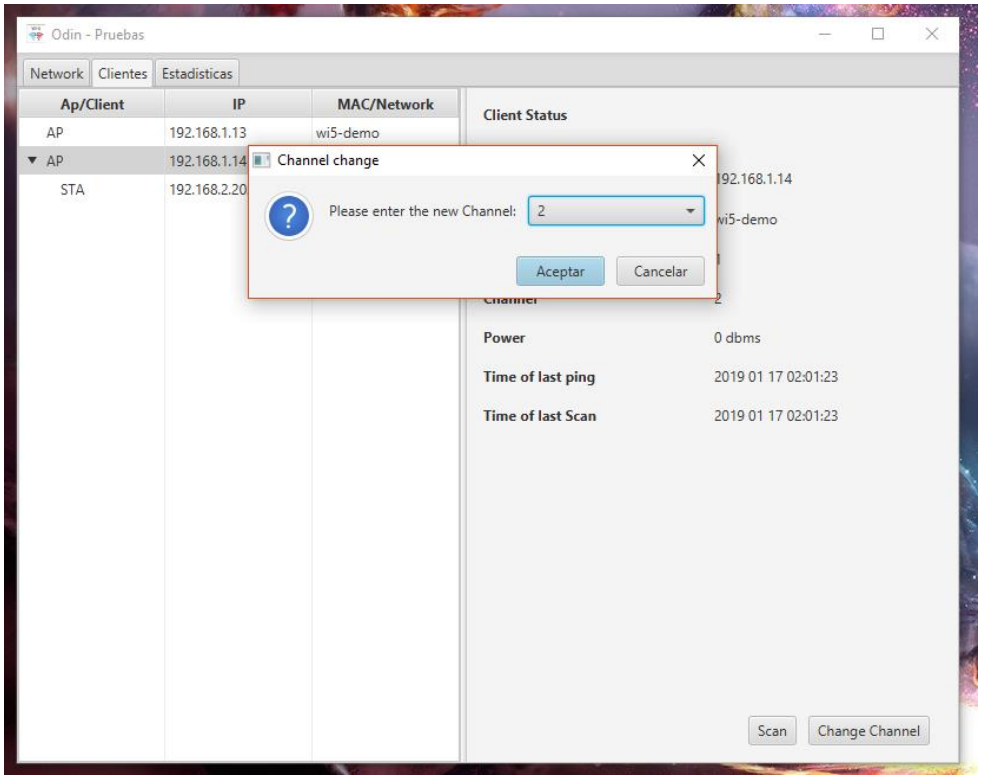

Figura 3.12. Ventana emergente *solicitud de cambio de canal*.

En caso de que el usuario solicite un escaneo, y el agente esté ocupado realizando otro escaneo, como puede ser por ejemplo el escaneo periódico realizado por la aplicación *Smart AP Selection,* se mostrarán datos de escaneos anteriores.

El segundo botón mostrará un desplegable donde el usuario puede elegir el canal al cual desea que el punto de acceso se mueva. Se omite el canal 14 debido a que es un canal reservado.

En caso de pulsar una estación en la tabla de la izquierda, se genera un evento que actualiza la vista que ocupa el lado derecho, ocultando diferentes etiquetas y modificando el comportamiento de los botones de abajo a la derecha.

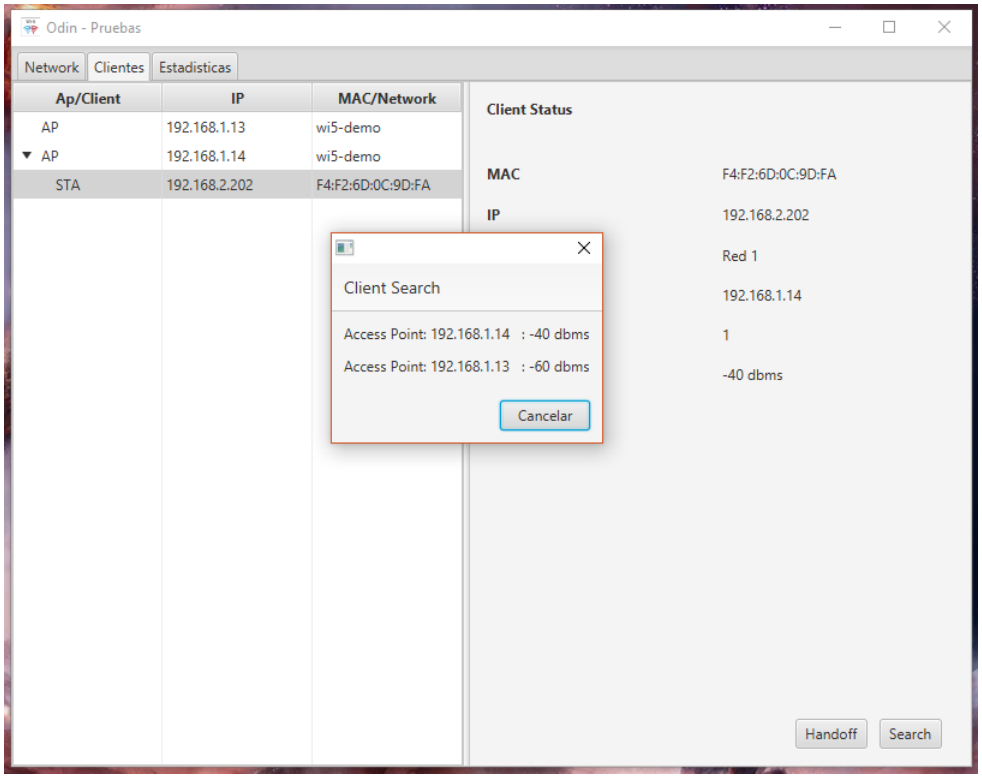

Figura 3.13. Ventana emergente de *solicitud de búsqueda de una estación*.

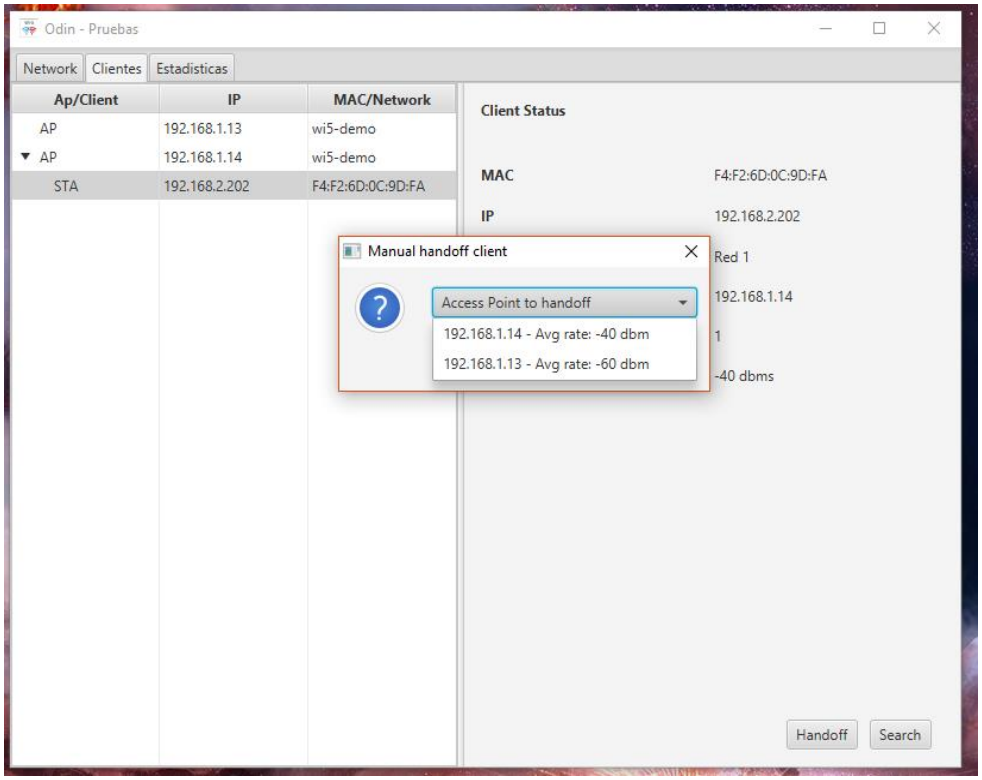

Figura 3.14. Ventana emergente de *solicitud de handoff*

Al solicitarse el *handoff* manual de una estación, se muestra la conexión de la estación con cada uno de los puntos de acceso disponibles, de tal manera que el usuario puede tomar una decisión informada.

4.2.3 Gráficas de tiempo en el aire, RSSI suavizado y número de paquetes

En la última vista se muestran los datos del último escaneo. Los escaneos realizados por los puntos de acceso se realizan durante un período de tiempo, durante el cual los agentes miden el número de paquetes que envía una estación, el tiempo que ha tardado en enviar ese número de paquetes y la señal de atenuación con la cual se envía dichos paquetes. Las gráficas del cliente reflejan estos datos.

Figura 3.15. Gráficas de las estaciones.

El usuario puede escoger el punto de acceso en el desplegable superior, y los datos se actualizan en tiempo real. Los gráficos permiten al usuario escoger más de un punto de acceso, permitiéndole comparar los datos mostrados. También se ha incluido un botón que permite borrar las gráficas.

Capítulo 5 - Organización y gestión del proyecto

5.1 Planificación del proyecto

El proyecto se planificó en un total de cinco etapas, tal y como se muestra en la Fig. 5.1, hasta alcanzar al momento final de entrega del producto. Estas cinco etapas parten desde el inicio del proyecto, contando desde la elección del tema del TFG hasta la entrega del proyecto finalizado al cliente, lista para la puesta en producción y conjuntamente a la documentación acerca del proyecto.

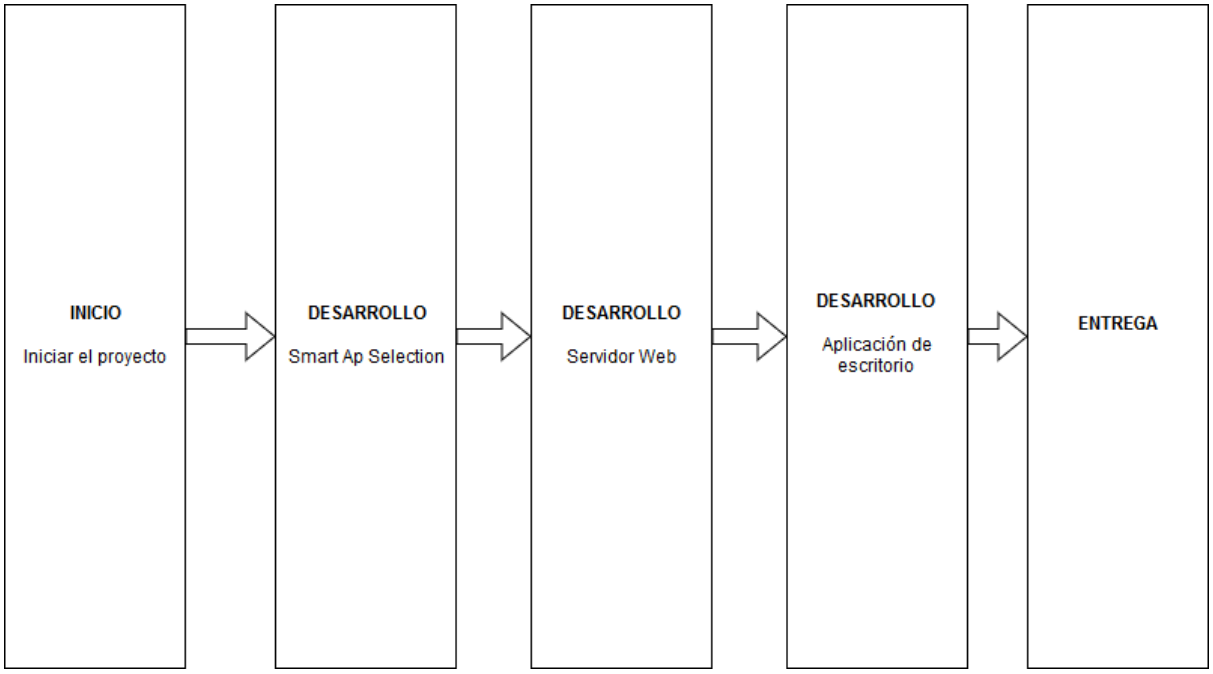

Figura 5.1: Ciclo de vida de desarrollo del proyecto.

5.1.1 Inicio

Durante esta fase se han estudiado las diferentes opciones disponibles para realizar un proyecto que pueda ser utilizado como Trabajo Fin de Grado. Una vez elegido este proyecto, el estudiante aborda los problemas que propone el cliente e intenta alcanzar unos requisitos finales (Sección 2.5) acordes con las necesidades del cliente. Se eligen las tecnologías que se consideran adecuadas y se propone una prueba de concepto (Sección 2.3.3) donde se realiza una aproximación para comprobar la viabilidad del proyecto. Las conclusiones obtenidas de este primer análisis se reflejan en el análisis y requisitos del sistema (Capítulo 2).

5.1.2 Desarrollo

Todas las fases de desarrollo cuentan con diferentes subfases como configuración, codificación y pruebas. Estas fases son las que mayor carga de trabajo engloban, ya que se entiende que las tareas a realizar pueden llevar a modificaciones sobre lo recogido en el Capítulo 2.

Como se aprecia en la Fig. 5.1, se ha subdivido la fase de desarrollo en tres partes diferenciadas, según se iban enfrentando los diferentes problemas, iniciando la fase de desarrollo por la actualización y refactorización del código de la aplicación *Smart AP Selection*, pasando por el servidor web y por último el desarrollo de la aplicación de escritorio.

5.1.3 Entrega

Una vez obtenida una solución que se considera entregable y que satisface los requisitos definidos, esta versión pasa las pruebas de aceptación llevadas a cabo por el cliente, en este caso el grupo de investigación.

A partir de esta versión, el grupo de investigación seguirá avanzando en el proyecto añadiendo nuevos desarrollos o funcionalidades que pueden ser puntos de partida para futuros Trabajos Fin de Grado.

En esta fase se obtiene toda la documentación que se ha ido desarrollando a lo largo del proyecto y conjuntamente a ésta se redacta la memoria que sirve para realizar el depósito del Trabajo de Fin de Grado.

5.2 Gestión del proyecto

5.2.1 Sistema y recursos

El desarrollo del TFG se ha realizado sobre un ordenador portátil con sistema operativo Windows 10 x86_64, con 4 Gb de Ram y procesador Intel Core i5-2430M. En este sistema se instalaron los diferentes entornos de desarrollo necesarios. Se ha utilizado *Eclipse IDE* 2018-09 [22] para el desarrollo de la aplicación cliente con *JavaFx*, y *IntelliJ IDEA Community* version 2017.2 [23] para el desarrollo del servidor *ApiRest* local con el que realizar las pruebas. Para el lanzamiento y conexión con los diferentes elementos del sistema se ha utilizado la aplicación *MobaXterm* versión 10.5 [24] y *Sublime Text* 3 [25] con una extensión *STFP* para programar en remoto.

El sistema y los elementos de red necesarios para el desarrollo de este proyecto han sido proporcionados por el grupo de investigación. Dicho sistema se compone de los siguientes elementos:

- Un *Intel NUC16 5i3RYH* (Fig 5.2), que cuenta con un procesador *i3-5010U, 16 GB* de memoria y disco duro *SSD*. Con sistema operativo *Vmware [26]*, el cual permite utilizar un equipo físico como servidor de virtualización. Las máquinas virtualizadas (controlador y router *DHCP* (*Dynamic Host Configuration Protocol*)) usan *Debian [27]* como sistema operativo, permitiendo crear un entorno estable y compatible.
- Dos puntos de acceso dispuestos a lo largo de la planta 2 del edificio Ada Byron, un *Netgear R6100* y un *TP-Link AC1750v2* (Fig 5.3).

Figura 5.2. Intel Nuc sobre el concentrador segmentado utilizado.

Figura 5.3. TP-Link AC1750v2 usado en el proyecto

5.2.2 Composición de la red

Para el desarrollo del proyecto el grupo de investigación dispuso una red privada donde llevar a cabo las pruebas y el desarrollo.

La red se compone de los elementos mencionados anteriormente y tiene la siguiente topología:

- Una red de datos compuesta por un router, un servidor DHCP y los dos puntos de acceso, que está conectada directamente a Internet y es el lugar por donde se comunican las estaciones.

- Una red de control compuesta por el mismo router, los dos puntos de acceso y el controlador donde se aloja la aplicación *Smart AP Selection*.

Los dos puntos de acceso se dispusieron en la planta tal y como se muestra en la figura, permitiendo realizar las pruebas moviéndose únicamente por el pasillo.

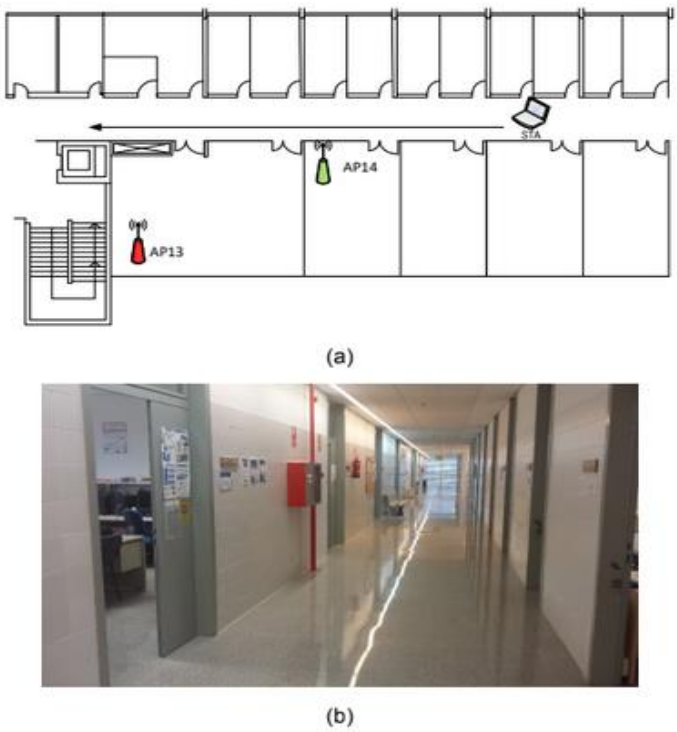

Figura 5.4. Composición de la red sobre el mapa del piso 2 del edificio Ada Byron.

5.2.3 Gestión de documentos

La gestión de documentos del proyecto se ha llevado a cabo haciendo uso de la aplicación *Google Drive,* donde se gestiona las diferentes versiones de la memoria, las imágenes, diagramas y el control de esfuerzos.

5.3 Esfuerzos

En la Figura 5.5 se refleja, de manera global, el cómputo de horas que realmente se han utilizado para cada una de las fases del proyecto. Esto ha sido posible gracias a que durante toda la realización del proyecto se ha llevado una contabilidad de las horas utilizadas en cada tarea a partir de anotaciones internas. Las horas consumidas para reuniones con el director del proyecto y las horas dedicadas a la redacción de la memoria del proyecto se han incluido dentro del cómputo de horas de las fases del proyecto.

Concepto	**Horas invertidas**
Análisis del sistema	80 horas
Diseño del sistema	110 horas
Implementación y desarrollo	145 horas
Total	**335 horas**

Figura 5.5 Cómputo global de horas

Capítulo 6 - Conclusión

Durante el desarrollo de la aplicación para la distribución del tráfico en un entorno WLAN se han adquirido ideas y conocimientos que pueden considerarse como conclusiones válidas para este proyecto.

Desde un inicio ha sido necesario considerar el alcance de este proyecto, ya que los requisitos propuestos por el grupo de investigación son demasiados ambiciosos para abordarlos en un único Trabajo Fin de Grado. Se ha realizado un sistema que pueda servir de base para futuros desarrollos que necesiten mostrar información obtenida a través de la aplicación *Smart AP Selection*. El proyecto se ha centrado en la creación de una aplicación de escritorio y un proceso de obtención/tratamiento de la información generada de manera periódica por la aplicación *Smart AP Selection*. De esta forma, el sistema desarrollado aporta una base para nuevas mejoras de la aplicación, ya sean desarrollados como TFG o como proyectos de desarrollo del equipo de investigación, aportando nuevas funcionalidades o expansiones del proyecto actual.

Una vez finalizado el proyecto, la aplicación *Smart AP Selection* ha evolucionado a una sistema robusto, funcional y mantenible que es capaz de obtener y mostrar la información almacenada en un sistema complejo como es Odin-Wi-5. La información se muestra ahora de manera amigable e incluso un neófito en la materia puede comprender el estado de situación de la red. Sobre la aplicación *Smart AP Selection* se ha realizado un "lavado de cara", siendo refactorizada con la idea de aumentar su rendimiento y mantenibilidad a lo largo del tiempo.

El trabajo realizado con las diferentes tecnologías usadas (*Java*, *JavaFx*, *FloodLight*, *VMWare*, entre otras) y la interrelación entre ellas ha concluido en un sistema robusto, capaz de cumplir con los objetivos y requisitos marcados en el inicio del proyecto, obteniendo una solución usable, atractiva, y mantenible, y que puede ser usado como base para siguientes proyectos, ya que ha sido diseñado con este objetivo.

A nivel personal, este proyecto ha sido un primer contacto con el desarrollo de un proyecto a nivel profesional de manera independiente, se han presentado problemas reales, a los cuales el estudiante ha tenido que aportar soluciones y alternativas, modificando en algunos casos la base misma del proyecto. También ha de tenerse en cuenta la experiencia positiva obtenida al haber trabajado en la línea de investigación iniciada con el proyecto europeo H2020 Wi-5 "*What to do With the Wi-Fi Wild West*".

Bibliografía

[1] F. Bouhafs, M. Mackay, A. Raschella, Q. Shi, F. den Hartog, J. Saldana, J. Ruiz-Mas, J. Fernandez-Navajas, R. Munilla, J. Almodovar, N. van Adrichem, "Wi-5: A Programming Architecture for Unlicensed Frequency Bands," IEEE Communications Magazine, vol. 56, no. 12, pp. 178-185, December 2018. doi: 10.1109/MCOM.2018.1800246

[2] El código de la aplicación se encuentra en https://github.com/Wi5/odin-wi5/wiki/Application-SmartApSelection, última visita enero 2019.

[3] Stapleton, J., DSDM Business Focused Development, 2nd ed. London, Addison Wesley, 2003

[4] L. Sequeira, J. L. de la Cruz, J. Ruiz-Mas, J. Saldana, J. Fernández-Navajas, J. L. Almodovar, "Building a SDN Enterprise WLAN Based On Virtual APs," IEEE Communications Letters, vol. 21, no. 2, pp. 374-377, Feb. 2017. ISSN 1089-7798. doi 10.1109/LCOMM.2016.2623602.

[5] Odin, disponibe en https://github.com/lalithsuresh/odin, último acceso enero 2019.

[6] J. Schulz-Zander, P. L. Suresh, N. Sarrar, A. Feldmann, T. Hhn, R. Merz, "Programmatic Orchestration of WiFi Networks.," in: G. Gibson, N. Zeldovich (Eds.), USENIX Annual Technical Conference, USENIX Association, 2014, p. 347358.

[7] Floodlight Project, disponible en http://www.projectfloodlight.org/, útimo acceso enero 2019.

[8] Aplicación Wi-5 SmartApSelection, ver https://github.com/Wi5/odin-wi5/wiki/Application-SmartApSelection,último acceso enero 2019.

[9] J. Saldana, R. Munilla, S. Eryigit, O. Topal, J. Ruiz Mas, J. Fernandez-Navajas, L. Sequeira," Unsticking the Wi-Fi Client: Smarter Decisions using a Software Defined Wireless Solution," in IEEE Access, vol. 6, pp. 30917-30931, 2018. doi: 10.1109/ACCESS.2018.2844088.

[10] O. Topal (Editor), J. Saldana, A. Raschella, Jan de Nijs, What to do With the Wi-Fi Wild West, Deliverable 5.2. Integration Results. Disponible en http://www.wi5.eu/wp-content/uploads/2015/02/D5.2-Integration-Results.pdf, último acceso enero 2019.

[11] Aplicación Wi-5 DemoStatistics, ver https://github.com/Wi5/odin-wi5/wiki/Application-DemoStatistics, último acceso enero 2019.

[12] Código de Smart AP Selection, disponible en https://github.com/Wi5/odin-wi5-controller/blob/master/src/main/java/net/floodlightcontroller/odin/applications/SmartApSelection.java, último acceso enero 2019.

[13] Martin, Robert C. "Design principles and design patterns." Object Mentor 1.34 (2000): 597.

[14] GraphStream project, disponible en http://graphstream-project.org/, último acceso enero 2019

[15] Puthalath, Lalith Suresh. "Programming the enterprise WLAN: an SDN approach." Instituto Superior Técnico (2012). Disponible en http://lalithsuresh.files.wordpress.com/2011/04/lalith-thesis.pdf, último accesoenero 2019.

[16] González, Y. D., & Romero, Y. F. (2012). Patrón Modelo-Vista-Controlador. *Revista Telem@ tica*, *11*(1), 47-57.

[17] Abego Treelayout Core, http://treelayout.sourceforge.net/, último acceso enero 2019.

[18] Kohler, Eddie, et al. “The Click modular router.” ACM Transactions on Computer Systems (TOCS) 18.3 (2000): 263-297.

[19] Aplicación Postman, https://www.getpostman.com/, último acceso enero 2019.

[20] Sistema de pruebas *JUnit,* https://junit.org/junit5/, último acceso enero 2019.

[21] Tecnología *Spring Boot,* http://spring.io/projects/spring-boot, último acceso enero 2019.

[22] Eclipse, https://www.eclipse.org/ide/, última vez abierto, enero 2019.

[23] IntelliJ IDEA, https://www.jetbrains.com/idea/ , última vez abierto, enero 2019.

[24] MobaXTerm, https://mobaxterm.mobatek.net/ , última vez abierto, enero 2019.

[25] Sublime Text 3, https://www.sublimetext.com/3 , última vez abierto, enero 2019.

[26] VMware, https://www.vmware.com, última vez accedido enero 2019.

[27] Debian, https://www.debian.org/index.es.html , última vez accedido enero 2019.

Anexos

Anexo 1 - Información que generan los puntos de acceso

Los puntos de acceso (AP) se comunican con el controlador indicando la calidad y parámetros de las múltiples conexiones simultáneas que puede tener con una estación. Los datos se resumen de la siguiente manera:

```
<<<<<<<<< Rx statistics >>>>>>>>>
Uplink station    MAC: 30:07:4D:9E:8E:C8
IP: 192.168.2.215
            num packets: 4175
            avg rate: 51261.7964072 kbps
            avg signal: -30.5969219499 dBm
            avg length: 295.96239521 bytes
            air time: 213.734666667 ms
            init time: 1519733522.191784021 sec
            end time: 1519733574.866460499 sec
<<<<<<<<< Tx statistics >>>>>>>>>
Downlink station  MAC: 30:07:4D:9E:8E:C8
IP: 192.168.2.215
            num packets: 3849
            avg rate: 54000 kbps
            avg signal: 10 dBm
            avg length: 702.74564822 bytes
            air time: 400.721185185 ms
            init time: 1519733525.415637841 sec
            end time: 1519733574.344232353 sec
```

Los puntos de acceso diferencian los paquetes enviados desde la estación al punto de acceso (Uplink) y los enviados desde el punto de acceso a la estación (Downlink).

Los puntos de acceso pueden poseer estadísticas de recepción (*Rx Statistics*) de estaciones no asociadas a la red. Mientras que sólo se generan estadísticas de emisión (*Tx Statistics*) si el cliente se encuentra asociado.

Anexo 1 - Información que obtiene la aplicación *DemoStatistics*

La aplicación *DemoStatisctics* muestra por pantalla diferentes estadísticas del sistema, he aquí algunos ejemplos:

Estadísticas internas obtenidas en la interfaz Wi-Fi de un punto de acceso (AP):

```
=======Internal statistics=======
Agent [0]: /192.168.1.13
      Txpower: 10 dBm
      Channel: 1
      Last heard: 389 ms ago

Agent [1]: /192.168.1.15
      Txpower: 10 dBm
      Channel: 6
      Last heard: 1893 ms ago

Agent [2]: /192.168.1.14
      Txpower: 10 dBm
      Channel: 6
            Last heard: 142 ms ago
```

Estadísticas de una única estación (STA):

```
<<<<<<<<< Rx statistics >>>>>>>>>
Uplink station    MAC: 30:07:4D:9E:8E:C8 IP: 192.168.2.215
            num packets: 4175
            avg rate: 51261.7964072 kbps
            avg signal: -30.5969219499 dBm
            avg length: 295.96239521 bytes
            air time: 213.734666667 ms
            init time: 1519733522.191784021 sec
            end time: 1519733574.866460499 sec
<<<<<<<<< Tx statistics >>>>>>>>>
Downlink station  MAC: 30:07:4D:9E:8E:C8 IP: 192.168.2.215
            num packets: 3849
            avg rate: 54000 kbps
            avg signal: 10 dBm
            avg length: 702.74564822 bytes
            air time: 400.721185185 ms
            init time: 1519733525.415637841 sec
            end time: 1519733574.344232353 sec
```

Ejemplo de estadísticas internas obtenidas mediante un escaneo activo con cuatro puntos de acceso (AP):

```
========Active scanning (Matrix of "distance in dBs")========
Agent to send measurement beacon: /192.168.1.7
      AP MAC: 14:CC:20:AC:72:91
      avg signal: -31.3936258323 dBm
      AP MAC: 14:CC:20:AC:72:91
      avg signal: -20.6498744445 dBm
      AP MAC: 14:CC:20:AC:72:91
      avg signal: -30.1345598285 dBm
Agent to send measurement beacon: /192.168.1.1
      AP MAC: 18:D6:C7:86:96:D2
      avg signal: -30.1322826573 dBm
      AP MAC: 18:D6:C7:86:96:D2
      avg signal: -30.1322826573 dBm
      AP MAC: 18:D6:C7:86:96:D2
      avg signal: -30.139743705 dBm
Agent to send measurement beacon: /192.168.1.3
      AP MAC: 18:D6:C7:86:97:BC
      avg signal: -21.1163317623 dBm
      AP MAC: 18:D6:C7:86:97:BC
      avg signal: -30.2889719889 dBm
      AP MAC: 18:D6:C7:86:97:BC
      avg signal: -39.5785848496 dBm
Agent to send measurement beacon: /192.168.1.2
      AP MAC: 18:D6:C7:86:98:25
      avg signal: -30.065637695 dBm
      AP MAC: 18:D6:C7:86:98:25
      avg signal: -30 dBm
      AP MAC: 18:D6:C7:86:98:25
      avg signal: -30.3385826726 dBm
============================================================

192.168.1.7 ----------  -31.39 dBm  -20.64 dBm  -30.13 dBm
192.168.1.1 -30.13 dBm  ----------  -30.13 dBm  -30.13 dBm
192.168.1.3 -21.11 dBm  -30.28 dBm  ----------  -39.57 dBm
192.168.1.2 -30.06 dBm  -30 dBm     -30.33 dBm  ----------
```

Ejemplo de estadísticas obtenidas mediante un escaneo pasivo:

```
==============Passive scanning (Matrix of RSSI)==============
AP /192.168.1.7 in channel: 3
AP /192.168.1.1 in channel: 1
AP /192.168.1.3 in channel: 11
AP /192.168.1.2 in channel: 6
============================================================
[ AP7 ][ AP1 ][ AP3 ][ AP2 ] - RSSI [dBm]
[-50,00][-50,00][-40,00][-99,90] Client /192.168.2.202 in agent
/192.168.1.3
============================================================
```

Anexo 2 - Modos de la aplicación *Smart AP Selection*

La aplicación *Smart Ap Selection* se puede ejecutar en diferentes modos, estos modos se indican en los parámetros de inicialización de la aplicación. Los modos son los siguientes:

Modo *RSSI*

El modo RSSI decide si se produce el *Handoff* de estación (STA) cuando estas tres condiciones se cumplan:

- La estación está al alcance de un punto de acceso con mejor *RSSI.*
- El tiempo de Hystheresis de la estación ha pasado.
- Se ha alcanzado el *Threshold* mínimo.

Modo *FF* (*Fittingnes Factor)*

En este modo el controlador moverá la estación al punto de acceso cuyo *Fittingnes Factor* sea el máximo, respetando el límite del *Threshold.* La función principal se ejecutará tal y como se muestra en la Fig 7.1.

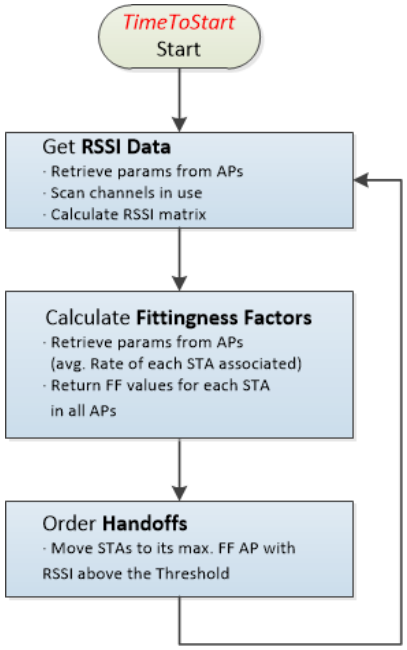

Figura 7.1

Modo *BALANCER*

El modo *BALANCER* garantiza una buena conexión (*RSSI > Threshold)* y al mismo tiempo el número de estaciones asociadas a cada punto de acceso se mantendrán en la media. Si el sistema tiene 20 estaciones asociadas, dentro de lo posible, cada punto de acceso dará servicio a 10 de ellas.

El bucle de la función principal seguirá los siguientes pasos:

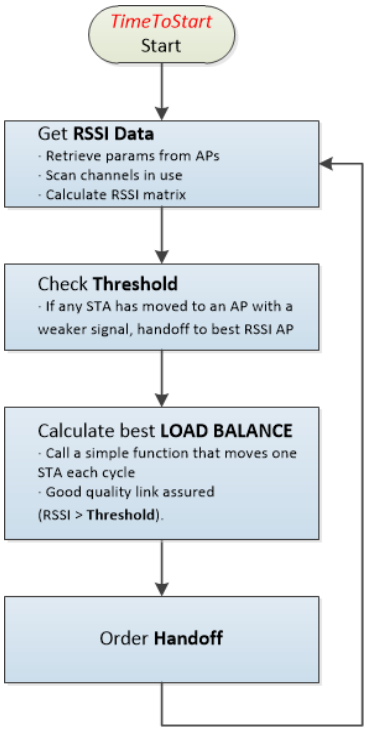

Figura 7.2

Modo *DETECTOR*

La aplicación *Smart Ap Selection* en este modo puede detectar un tipo de flujo de información en particular. Si detecta ese flujo entre una de las estaciones y uno de los puntos de acceso, moverá la estación a un punto de acceso “VIP”. Una vez dicho flujo finaliza, por ejemplo, se termina una conexión *ssh*, el controlador mueve la estación a su anterior punto de acceso.

El bucle de la función principal será el siguiente.

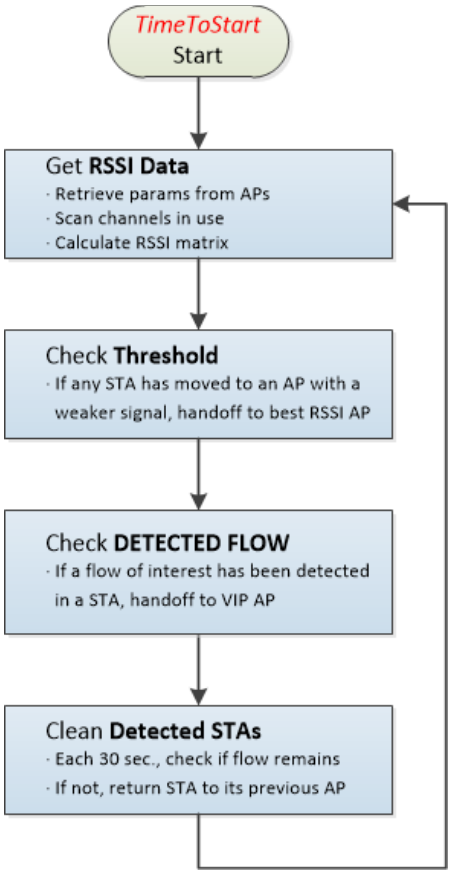

Figura 2.3